ENSURING PEDESTRIAN SAFETY ON CAMPUS THROUGH THE USE OF COMPUTER VISION

A Dissertation
Presented to
The Academic Faculty

by

Domitille Commun

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
May 2019

ENSURING PEDESTRIAN SAFETY ON CAMPUS THROUGH THE USE OF COMPUTER VISION

Approved by:

Dr. Dimitri Mavris, Advisor
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Jeffrey T. Hunnicutt,
Physical Security Specialist
Georgia Tech Police Department

Dr. Olivia Pinon Fischer,
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Dr. Michael Balchanos,
Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: April 19, 2019

ACKNOWLEDGEMENTS

I would like to thank my advisor Doctor Mavris for his guidance while I am completing my degree. I would like to thank him for giving me the opportunity to be a member of ASDL and to work on very challenging and interesting projects while I am at Georgia Tech. Working on research at ASDL has been my favorite academic and professional experience so far. I have learnt a lot about conducting research and about aerospace engineering since I arrived in the lab. Thank you very much Doctor Mavris for giving me the chance to live this experience.

I would like to thank my advisors Doctor Olivia Fischer and Doctor Balchanos for their recommendations and help throughout this thesis. Thank you for your time and for the improvement you helped me to do in this thesis.

I would like to thank Jeffrey Hunnicutt from the Georgia Tech Police Department for sharing his knowledge about pedestrian safety on the Georgia Tech campus and helping me to get an accurate perspective of the work that could be achieved. I would like to thank him for providing me video recordings of intersections on campus. Without these data I couldn't have achieved the work I have done in this thesis. Thank you very much Jeff.

I would like to thank my family, my friends and lab mates who supported me during this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In the United States alone, 5987 pedestrians were killed and 70,000 injured in 2016 and 2015, respectively. Those numbers are of particular concern to universities where traffic accidents and incidents represent one of the main causes of injuries on campuses. On the Georgia Tech Campus, the growth of the population-to-infrastructure ratio, the emergence of new modes of transportation, and the increase in the number of distractions have shown to have an impact on pedestrian safety.

One means to ensure safety and fast responses to incidents on campus is through video surveillance. However, identifying high risk situations from video cameras and feeds require significant human efforts. Computer vision and other image processing methods applied to videos may provide the means to reduce the cost and human errors associated with processing images. Computer vision in particular provides techniques that enable artificial systems to obtain information from images. Hence, while humans primarily use their eyes and brain to understand their surroundings, cameras can be used as eyes and computers can be used as brains to reach similar capabilities. However, while humans understand information from their surroundings, computers see images as arrays of pixel intensity. While many vendors provide computer vision and image recognition capabilities, additional efforts and tools are needed to support 1) the mission of the Georgia Tech Police Department and 2) the identification of solutions or practices that would lead to improved pedestrian safety on campus.

Data from cameras can be systematically and automatically analyzed to provide situational awareness and support enforcement operations:

- Characterize conflict situations including pedestrians.
- Provide calibration data to optimize traffic light control, as this would contribute in reducing jaywalking and improving overall travel time.
- Automatically issue citations.
- Document police reports.
- Automate or better inform enforcement operations.

In particular, this research aims at developing an intelligent system that will automate data collection on incidents occurring around campus and attempt to optimize traffic light control.

This is achieved by:

1) Leveraging computer vision techniques such as object detection algorithms to identify and characterize conflict situations including pedestrians. Computer vision techniques were implemented to detect and track pedestrians and vehicles on surveillance videos. Once trajectories were extracted from videos, additional data such as speed, vehicle and pedestrian flows and collisions were determined. Such data can be used by the Georgia Tech Police Department to determine needs for agents to manage traffic at a given intersection. Speed information is used to detect speeding automatically, which can help to enforce law in an automated way.

Traffic and walking light color detection algorithms were implemented and combined with location data to detect jaywalking and red light running.

The conflict situations detected were stored in a database which completes the Police record database. The data is structured such as to enable to get statistics or the detection of patterns without a lot of processing.

Hence, the tool built in this thesis provides structured information about violations and high risk situations around campus. This data can be used by the Police Department to automate law enforcement and issue citations automatically and to determine the needs for countermeasures to ensure pedestrian safety.

2) Implementing a simple optimized traffic light control and setting up the inputs necessary for a better optimized traffic light control using reinforcement learning.

It is expected that the improved situational awareness and information gained from developing these capabilities will contribute to help reduce the number of collisions, the amount of dangerous jaywalking and lead to new ways to ensure pedestrian safety on campus.

CHAPTER 1. INTRODUCTION AND MOTIVATION

In the United States 5987 pedestrians were killed and 70,000 pedestrians were injured in traffic crashes in 2016 and 2015, respectively. [1] The U.S. population is increasing by about 0.71 percent per year [2], consequently the number of pedestrian fatalities and injuries is expected to increase as well. In order to limit this increase, new measures and methods that seek to improve pedestrian safety should be implemented.

University campuses are concerned by this safety problem. This concern is especially due to the fact that accidental injuries are the main cause of death among American college students, with motor vehicle accidents being the main cause of these accidental deaths. [3] Implementing new measures to improve driver behavior and pedestrian safety should improve safety on campus.

1.1 Pedestrian safety in the US

The number of pedestrian fatalities in the United States has not decreased significantly since 1995. While 5584 fatalities occurred in 1995, 5376 fatalities were reported in 2015 in the United States [1].

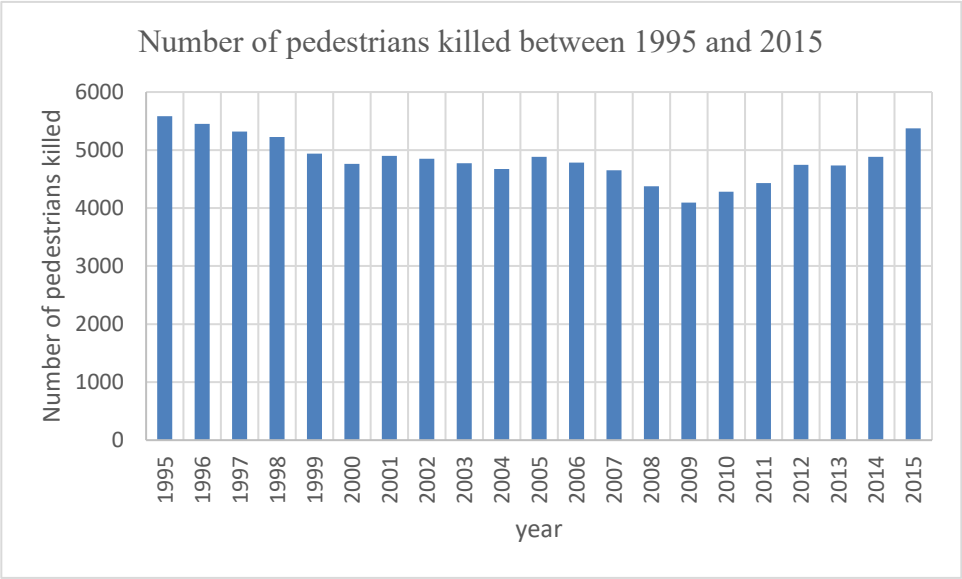


Figure 1: Number of pedestrians killed between

The number of pedestrian injuries decreased by 18.6 percent between 1995 and 2015, however 70,000 pedestrians were still injured in traffic crashes in 2015 [1]

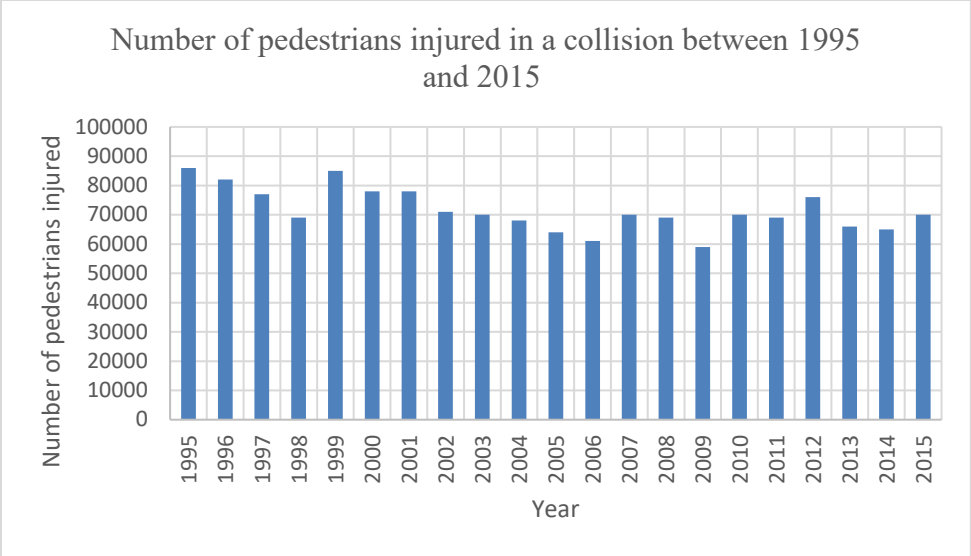


Figure 2: Number of pedestrians injured in a collision between 1995 and 2015 [1]

Pedestrian safety is a growing concern on university campuses. As mentioned previously, vehicle accidents are one of the main causes of death among college students. A study was conducted on an American university campus and showed that the most common injury circumstances occur at intersections involving a driver turning or failing to stop at a red light. [4] Moreover, in the United States, 41 percent of crashes including pedestrians occur at intersections [5]. With this large portion of crashes, and especially injuries occurring at intersections, focusing on improvements at intersections is a relevant strategy to improve pedestrian safety.

1.2 Pedestrian safety on the Georgia Tech campus

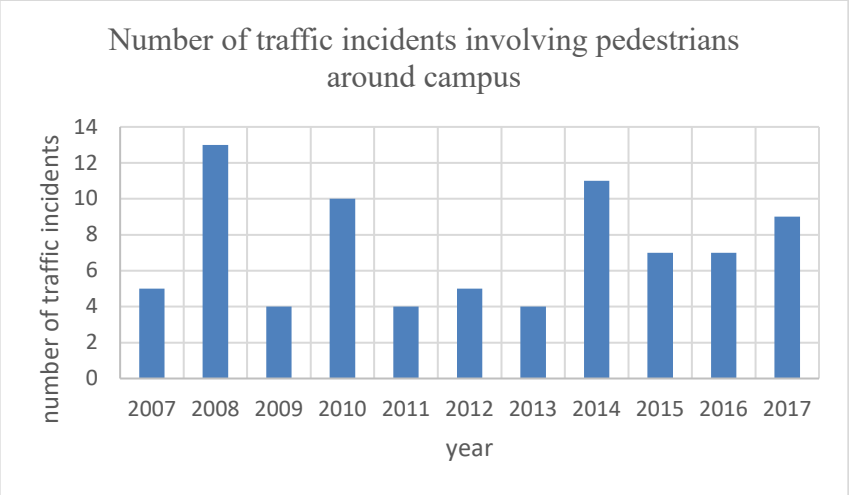


Figure 3: Number of traffic incidents around campus involving pedestrians

On the Georgia Tech campus, the number of pedestrian injuries per year reported between 2006 and 2017 varies from 4 (reached in 2009, 2011 and 2013) to 13 (reached in 2008) according to the Georgia Tech Police record database. Last year, in 2017, 9 pedestrian injuries were reported on campus.

In the meantime, the population has grown rapidly on campus. The number of students, faculty members, and staff members increased from 27,539 to 37,222 between 2012 and 2017. [6] The increase in the number of students enrolled is the main driver of the campus population growth. 21,558 students were enrolled at Georgia Tech in Atlanta in 2012 and 29,369 students were enrolled in 2017, which means that the enrollment has increased by 36.2 percent between 2012 and 2017.

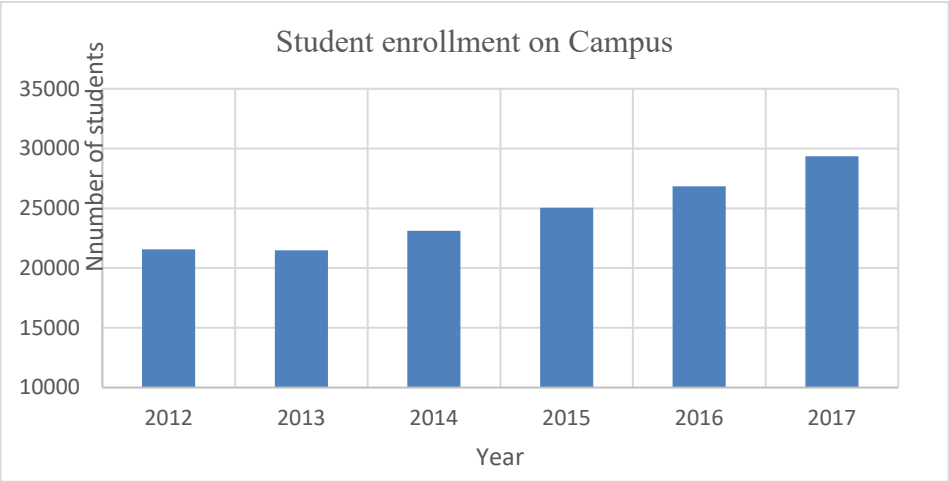


Figure 4: Enrollment growth on campus

While the population has been increasing on campus, the campus infrastructure has remained static for the most part, thus leading in an increase in traffic and pedestrian flow through intersections on campus. Consequently, there is a higher likelihood for pedestrian injuries to occur.

In addition to the increasing population-to-infrastructure ratio, two emerging trends are impacting pedestrian safety on campus. First, additional modes of transportation have been emerging and can represent a threat for pedestrian safety. The proliferation of electric scooters is an example of such emerging mode of transportation. More than 16 million rides have been recorded since the two main companies providing this service launched scooters in the United States last year. This new external factor represents a risk for pedestrian safety as most riders use sidewalks and the speed of a scooter can reach 14.8 miles per hour (mph). At this speed, there is 10% of chance of severe injury when a

pedestrian is hit by the transport system (scooter). [5] Additionally, it is hard for those driving vehicles on the road and for pedestrians to predict the behavior of someone on a scooter who sometimes may behave as a pedestrian and sometimes may behave as a vehicle. Thus there is more risk for pedestrian safety due to new modes of transportation.

Secondly, the number of distractions for pedestrians and drivers has increased significantly during the recent years with more distraction implying additional risk for pedestrian safety. As an example, the number of mobile phone applications available in the Google Play store reached 3,500,000 in 2017 while it was less than 10,000 in 2009. [7] The number of mobile phones in the US is also increasing and consequently more drivers are distracted while driving and more pedestrians are distracted by conversations and other uses of mobile phones while crossing streets. [8] With the increased use of mobile phones, the frequency of close calls (a narrow escape from danger) and the time to cross intersections also increase [9], leading to additional risk for pedestrian safety. A study conducted on an American university campus confirms that a large proportion of students are using headphones or handheld devices in proximity to vehicles and intersections. [4]

Pedestrian safety on the Georgia tech campus is a growing concern, as shown by the three key drivers identified. These drivers include the population increase on campus, the proliferation of new transportation systems, and the increase in the types of distractions. These driving factors, which challenge pedestrian safety, are not temporary, ie the expectation is that these trends will still be observed in the future. Past and current risks,

and their expected increase, raise the importance of identifying new ways to ensure pedestrian safety today and in the future.

In addition, as seen previously, pedestrian injuries and fatalities do not only concern university populations, they are now becoming a worldwide problem. Improving pedestrian safety on the Georgia Tech campus represents a good starting point towards addressing and improving pedestrian safety in the United States and worldwide. Indeed, finding solutions at the campus scale will provide insights that may improve pedestrian safety at a much larger scale.

1.3 High risk situations for pedestrians

The present section provides a description of "high risk situations" encountered by pedestrians. First, the term "conflict" must be defined: a *conflict* is an observable situation in which two or more road users approach each other in time and space to such an extent that there is risk of collision if their movements remain unchanged [10]. The risk of conflict must be determined for an area to determine what countermeasures are needed to improve pedestrian safety. However, conflict risk estimation is challenging, as crash data are often limited. One means however, to compute conflict risk is through safety surrogate indicators, as further discussed in Chapter 2.

As mentioned previously, a good strategy for pedestrian safety improvement is to start at busy intersections, where most injuries occur. "Conflict" includes all complex pedestrian-vehicle interactions, but at intersections, pedestrian and vehicle interactions are mainly described by the following situations: [5]

- Driver yielding too late to pedestrians at crosswalks.
- Speeding by vehicles at crosswalks.
- Vehicles stopping too close to crosswalks.
- Pedestrians stepping into traffic.
- Pedestrians or vehicles crossing against what the light signal indicates.

1.4 Problem statement

The population on campus continues to increase while the infrastructure remains static; indeed the infrastructure expands at a much slower rate than population. Consequently, the traffic and pedestrian flow are increasing at intersections representing a risk for pedestrians. Additional external factors such as new modes of transportation and new distractions for drivers and pedestrians also threaten the safety of pedestrians. Consequently, it is important to implement new methods and take new countermeasures to reduce risk for pedestrians. Before taking measures it is necessary to select the right location and the right countermeasures. [11] With that in mind it is necessary to get data about the frequency, type, and the location of high risk behaviors.

Research Objective: Describe, detect and quantify the frequency of high risk behavior and near-accidents conditions involving pedestrians in order to explore, assess and recommend traffic countermeasures to improve pedestrian safety.

CHAPTER 2. BACKGROUND

In order to ensure pedestrian safety communities are working with police departments to implement enforcement measures such as red light running detections or speeding enforcement. Other measures gathered under the title of “traffic calming measures” are employed by communities to ensure traffic safety.

Before implementing them, data must be obtained to identify the needs for countermeasures. A promising way to get such data is through the use of surveillance videos.

The chapter first introduces the current measures employed by communities to ensure pedestrian safety. Then it introduces the ways to get data about pedestrians before choosing a traffic treatments. In a second time this chapter introduces the use of surveillance cameras as a promising tool to ensure pedestrian safety.

2.1 Current measures to ensure pedestrians safety

2.1.1 Educational, engineering and enforcement operations

Today, communities employ *traffic calming* as a measure to improve pedestrian safety. Traffic calming consists of educational, engineering, and enforcement measures which aim to ensure pedestrian safety and encourage driver to respect traffic rules. [12]

Educational measures

Children is the primary demographic that is targeted by most traffic calming measures. Parent education seminars, classroom safety posters or school safety flyers that highlight pedestrian safety measures are examples of educational measures.

Engineering measures

Engineering measures consist in using devices such as speed humps, pedestrian refuges, or chicanes to alter the geometry of the street. The aim is to reduce the negative effect of vehicle use. Another measure that can be cited is *traffic light regulation*. Indeed, an optimal traffic light system reduces the risk of jaywalking, if it takes the pedestrian waiting time into account. [4] In that way, controlling traffic lights contributes to pedestrian safety.

Enforcement measures

A third measure is implemented by Police Departments who select different locations and times based on traffic accident data analysis, community inputs, and officer observations to enforce speed limits and driver yielding. This last measure requires significant police resources. Indeed, the most common way to implement enforcement operations requires one or several spotter(s) who observe the traffic and record violations that are transmitted to uniformed officers. These officers react by stopping the vehicle and issuing citations. [5] Implementing enforcement measures leads to better driver behavior, especially an increase in driver yielding, however it requires significant police resources. To make these operations more efficient, some optional tools can be used such as those that determine distance, markers such as traffic cones, Radar or LIDAR to record speed, or

video cameras to record infractions. [13] All these enforcement operations require lots of resources. A state-of-the-art approach to enforce speed limits and stopping at red-lights was developed using electronic devices to detect violations and provide photo evidence of infractions. This state-of-the art method is called *automated enforcement* and uses mainly two kinds of cameras: Red-Light cameras (RLC), which are used to prevent the running of red lights and automated speed enforcement (ASE) cameras, which are used to prevent speeding.

Red Light running enforcement

Red-light Running crashes are responsible for 260,000 injuries and 750 fatalities each year. [14] Red-Light Running Cameras (RLC) are combined with sensors such as inductive loops to detect violations. The inductive loops are located before the stop sign line and measure the speed of vehicles to predict whether the vehicle will be able to stop on time or not. Following this result, pictures of the vehicle are taken by the red-light camera. [15]

Automated speed enforcement

Automated Speed Enforcement (ASE) operations are combined with radar or LIDAR for speed detection before taking pictures of the violation. For both cameras, once pictures of traffic violations are taken, a citation can be sent automatically, or after a manual check. [15]

Automated enforcement operations can further assist police officers for traffic calming operations, while reducing resources required during operations and allowing for continuous enforcement. Moreover, they have proved to be effective worldwide [16]. In Chicago a reduction of 73 percent of dangerous behaviors has been seen where automated enforcement has been implemented. In Washington DC, traffic fatalities dropped from 68 to 19 between 2003 and 2012 where 25 ASEs have been set up. In Norway, automated enforcement systems have enabled the reduction of traffic fatalities and injuries by 20 percent. In London, traffic related deaths and serious injuries have decreased from 68 to 29, and from 813 to 593 respectively in two years thanks to ASE and RLC.

Despite their many benefits, these state-of-the art methods have limitations: they require sensors that are expensive and require significant time investment to set up. Using cameras without the use of sensors for automated enforcement would help eliminate this drawback.

A different approach using "average speed cameras" enables speeding determination while avoiding the expensive sensors (such as inductive loops). [17] Average speed cameras use several cameras along a road separated by a known distance and record each car's license plate passing in front of these cameras. Then the average speed is computed using the time difference between the moment the car is in front of camera A and the moment it is in front of camera B, and the known distance between the cameras. However, this method requires cameras installed at known distances along the same road and this method cannot be applied around the Georgia Tech Campus.

A new system using cameras only was developed for speeding and red light running detection; however, it requires the use of three specific cameras that are not police surveillance cameras and whose purpose is to detect these violations only. [18]

This leads to a first gap: There does not exist a means to conduct enforcement automatically using only surveillance cameras in a city environment.

As described in the section, three main measures are used by communities to ensure pedestrian safety: educational, enforcement and engineering measures. Before implementation, data must be collected to identify the needs for countermeasures.

2.1.2 Data collection before measure implementation

Before implementing traffic calming measures, communities must determine where these measures should be implemented. This is often accomplished using accident data, community inputs, such as resident's complaints, and local officers' observations. However, such data is often too limited to allow for the determination of accident estimates and countermeasures to be applied. Consequently, more detailed data about high-risk situations for pedestrians must be collected. Getting such data is expected to help better determine the needs for traffic calming measures and where they should be implemented.

Pedestrians and vehicles interactions at intersections are mainly characterized by vehicles delaying to yield to pedestrians at crosswalks, vehicle speeding at crosswalks, vehicles stopping too close to crosswalks, pedestrians unexpectedly stepping into traffic, pedestrians crossing against the walk signal and vehicles crossing against the traffic light

signal. [5] Obtaining data on recorded occurrences of such practices is key for determining the needs for countermeasures and develop suitable engineering solutions, or enforcement measures. A promising way for collecting this information is to use surveillance videos, since they record pedestrians and vehicles behaviors at intersections.

In addition to giving access to detailed data, cameras are state-of the art devices for enforcement and engineering operations because they enable the implementation of automated enforcement and provide real time traffic data that are required for traffic light control. However, as of present, surveillance cameras are mainly used to monitor communities from a centralized control center [19]. When a traffic issue is detected, the police will react by dispatching officers immediately. Hence, surveillance cameras are mainly used in a reactive way instead of a proactive one. Surveillance video cameras are not used to their full potential.

On the Georgia Tech campus, a team with GTPD is assigned to monitor a video wall with inputs from the security cameras. Two analytics tools are provided by the surveillance system, the first one detects any unusual event such as a bird landing on a roof where no birds usually land. The second one allows for the detection of a person that matches criteria such as hair color, clothes color etc. However; at the present time, GTPD has no capability that helps to detect abnormal behavior at intersections and the current analytics tools are used in a reactive way and not in a proactive one.

This is a second gap: Surveillance videos cameras and existing tools on campus are not used to inform risk to pedestrian safety.

A capability that leverages data from video surveillance cameras would offer a means to automate enforcement operations and to optimize engineering measures such as traffic light control.

The main research objective is to explore options on how surveillance videos can be better leveraged to support the development of an analytical tool that will help to improve pedestrian safety.

2.2 Use of surveillance videos

2.2.1 Common use of surveillance videos

Surveillance video cameras help to minimize response time and maximize operational effectiveness. Indeed, officers can be immediately deployed when certain situations arise that are detected using video cameras. For this reason, cameras help to reduce the number of incidents. Moreover, they improve situational awareness, which help officials manage and coordinate responses because they have real time access to visual information. In addition to improving effectiveness of operations, surveillance video cameras can be used to prove liability in disputes around accidents that may involve injury or property loss. [19]

As described previously, surveillance cameras are mainly used in a reactive way. This thesis proposes to analyze surveillance video data in real time to enable automated enforcement without sensors and real time traffic light control. To achieve this goal, data analytics on video must be performed.

2.2.2 Analytics on videos

Recent advances in Machine Learning (ML) techniques for object detection allow for automated analysis of surveillance videos. Analytics on videos can support reduction in human errors and make patrol operations more efficient, contributing towards reduced costs. While humans sense and understand activity within their surroundings, computers observe images as arrays of pixel intensity. Analytics on video must be done to interpret these inputs, understand images and provide accurate information about detected objects as output. *Computer Vision* provides techniques that enable building artificial systems that obtain information from images. Computer vision is the field of Artificial Intelligence (AI) that enables computers to observe, identify and process images in the same way as human vision does and then provide the appropriate output [20]. It is imparting human intelligence and instincts to a computer.

Currently, little use of computer vision capabilities have been done by enforcement agencies. [21] Law enforcement computer applications focused on data-mining and crime mapping and not on computer vision. The common computer vision algorithms enable facial recognition and license plate reading but the global use of computer vision for safety purpose remains under-explored. Some studies focus on congestion detection using cameras [22] or vehicle assistance systems such as parking assistance, rear-view detection, weather detection on cameras mounted on vehicles [23].

Today, by using computer vision techniques on videos, speeding, red light violations and license plate recognition can be automated. However this requires the use of specific cameras that are not video surveillance cameras. For instance, a combination of monochrome cameras, and single color camera in a stereo arrangement (several lenses are used to provide the ability to capture three-dimensional images) can be used to perform such detection. [18] In the context of this research, and as a mean to reduce costs for the Police Departments, cameras already installed on campus should be leveraged. The following paragraphs present some past studies then leveraged existing video surveillance cameras for incident detection.

A company called Traffic Vision has succeeded in detecting traffic incidents using existing cameras on highways without the need to retro-fit these cameras. However, their work focused on highway and vehicle traffic: congestion detection, wrong-way vehicle detection, slow speeds. Their work did not detect pedestrian incidents, which constitute a major difference with the work done in this thesis. Moreover, the cameras are located on highways, so the environment is not as complex to analyze as an urban environment which contain element such as trees or buildings that make the detections (traffic light color detection for instance) difficult. [23]

In this work, pedestrian hazardous situations must be detected and cameras located in cities must be used, which constitute a need compared to the current technologies.

Some works have focused on the analysis of pedestrian behavior through the use of video recordings. They can be divided into three methods. All of them start with a detection

and tracking of pedestrians and vehicles in videos. Then these steps are followed by the computation of location and speed.

- The first set of methods ([25], [26]) enables statistics about three hazardous situations for pedestrians: whether the pedestrians crossed in designated areas, crossed near moving vehicles or ran when crossing. The drawback of this method is that it doesn't provide any indicator about the level of danger of the situation.
- The second set of methods ([26] [28] [29]) enables to classify interactions as dangerous, safe or critical. However, the nature of incidents are not classified, so jaywalking or speeding or red light running are not clearly identified.
- The third set of method [30] enables the classification into three level of danger of vehicle-pedestrian interactions. It requires to compute indicators that depend on the direction of the pedestrian. However, this method does not take the vehicle speed into account, which is an important parameter at intersection.

None of these method focus on the following interactions: vehicles delaying to yield to pedestrians at crosswalks, vehicle speeding at crosswalks, vehicles stopping too close to crosswalks, pedestrians unexpectedly stepping into traffic, pedestrians crossing against the walk signal and vehicles crossing against the traffic light signal. However, it is necessary

to get information about these hazardous event to be able to determine the needs for countermeasures at intersections. [5]

This is a third gap: No methods have been implemented that enable statistics of the situations defined above.

The aim of this research is to add the following capabilities to computers: automated analysis and understanding of surveillance videos provided by campus cameras. These capabilities would allow for the detection, description and quantification of high risk situations for pedestrians and eventually lead to the identification of relevant countermeasures (traffic calming measures) and the development of suitable strategies to ensure pedestrian safety.

As part of the objectives of this research, an algorithm that extracts, analyses and understands useful information about pedestrian safety from videos will be built. These algorithms could be used to deploy enforcement operations and engineering measures such as traffic light control on campus in order to improve pedestrian safety.

The chart below describes the way the data extracted from videos could be used to improve pedestrian safety.

As mentioned in Chapter 1, the objective of this research is the following:

To detect, describe and quantify high risk situations and near accidents involving pedestrians by using computer vision techniques in order to explore, assess and recommend traffic countermeasures to improve pedestrian safety.

In the following chapter, the questions and hypotheses that this thesis attempts to answer will be presented. Then the approach proposed is provided in Chapter 4. Its implementation and the results are discussed in Chapters 5 and 6. Finally Chapter 7 provides some conclusions and avenues for the future work.

CHAPTER 3. PROBLEM FORMULATION

In order to improve pedestrian safety and determine the needs for countermeasures, data about high risk situations must be. This information collection is presented in the first section. The second section focuses on a measure to improve pedestrian safety: the traffic light color control.

3.1 Automate detailed data collection about high-risk situations using computer vision

This section discusses first the reasons for data collection and then presents the technical aspects of this data collection.

3.1.1 Need to collect detailed data about high risk situations involving pedestrians in an automated way

Before certain traffic measures are applied in an area to improve pedestrian safety, data collection procedures must be undertaken to evaluate high risk behaviors such as excess traffic and speeding. Consequently, there is a need to get data about high risk situations.

Three main methods enable the collection of pedestrian data [25]

- Pedestrian count: This method provides pedestrians volume and behavioral data that can provide insights into crash causes and can be used to determine certain areas where potential countermeasures are required. Most of the time the information is collected by the Department of Transportation and used to identify changes in pedestrian behaviors and trends, determine the

impact construction has on walking behavior and determine volume and repartition of non-motorized road users [25]. While pedestrian count provides local data on pedestrian volume and allows to obtain monthly estimates by scaling up information from a few observations, this method can be time consuming if done manually.

- Surveys: They are used to collect information about behaviors, views and experiences of the mainstream population. They provide data about the quality of the walking environment, pedestrian needs or concerns. Although they are inexpensive to analyze, the wording can bias the answers. Hence it can also be a challenge to obtain information at a high participation rate.

- Traffic records system: These systems gather crash data, roadway data, driver data, citations data, and injury data. Crash and roadway data are mainly provided through police reports that record crashes, vehicle and person-specific data about the parties involved. This data source is often subjective and lacks detail on incidents and what the situation was before the incident occurred. Driver data are obtained through the driver licenses and citation documents. These provide insight about the driver's behavior such as speeding or past incidents in which he/she has been involved. However, no pedestrian data is available from this source. Injury surveillance systems provide data about pedestrians after the incident occurs such as medical information.

Table 1 summarizes the advantages and limitations of each of these methods.

Table 1: Summary of the methods that enable the collection of pedestrian data

Method	Pedestrian count	Survey	Traffic record system
Data provided	Pedestrian traffic volume and directions	Quality of walking environment Pedestrian needs	Crash data, roadway data, citation data…
Advantage	Can be done automatically using electronic devices	Easy to analyze	Provide historical data
Drawbacks	Manual count is time consuming	Answer can be biased Difficult to get a high response rate	Lack of details on circumstances Subjective

Combining all previous data sources can provide accurate information that is useful in tracking pedestrian injuries and fatalities. However, there is a lack of details on incident circumstances and conditions at time of crash.

Consequently, additional data on traffic behavior at intersections, such as vehicle speeding at crosswalks or pedestrians crossing against the walk signal, is necessary [13]. Current methods to acquire pedestrian data do not provide the necessary information [30] [32] [27]. There is thus a need for a set of analytical methods that provide detailed information about unsafe situations in an automated way.

This leads to the first research question for this research:

RQ1: What approach(es) would allow for high-risk situations to be automatically detected?

As discussed, analyzing surveillance camera videos at intersections is expected to provide details and information about unsafe situations involving pedestrians such as speeding by vehicles, pedestrian crossing against the walk signal, vehicles yielding too late,

etc. Video recordings represent a promising and objective source of data about both pedestrian and driver behavior. When, combined with analytical techniques, video recordings are expected to provide the data necessary to implement countermeasures and help support automated enforcement operations.

While GTPD owns video recordings of campus intersections, they do not currently have the tool to analyze these videos in real time to get data about high-risk situations at intersections.

As discussed, the field of computer vision is expected to provide the necessary analytical techniques. [26] Such techniques include real-time object detection and tracking that enable the real-time analysis of video content. To be relevant and of use to law enforcement agencies, the set of techniques must have the characteristics discussed in the section below.

3.1.2 Technical capabilities needed

3.1.2.1 First set of information to be extracted from videos

The environment to be developed must be capable of extracting video captures of the following situations [5] , along with their corresponding metrics from video recordings:

- Speeding by vehicles. In particular, the vehicle speeds need to be recorded and compared to a speed limit.
- Vehicles stopping too close to crosswalks. In particular, the vehicle locations and crosswalk location need to be recorded and compared to a minimum safety distance.

- Pedestrian stepping into traffic. In particular pedestrian locations need to be recorded, and the vehicles and traffic light colors detected. Doing so would allow for the classification of situations as dangerous, or not.

- Driver yielding too late. In particular, the vehicle and pedestrian locations must be recorded to compute the vehicle-pedestrian distance. In addition, the vehicle speeds need to be recorded to determine whether the vehicles are able to stop on time so that neither vehicles nor pedestrians have to change their path to avoid a collision.

- Pedestrians or vehicles not respecting the traffic light signals. Traffic light colors must be detected. In addition, pedestrian and vehicle trajectories must be recorded and compared to cross areas when traffic lights are red. If pedestrians and vehicles were already crossing the street when the light turned red then the situation is not considered dangerous. If they started to cross the street after the light turned red, then the situation is classified as dangerous.

To summarize, the following parameters have to be extracted from videos: vehicles and pedestrian trajectories and speeds, traffic light colors, sidewalk locations.

3.1.2.2 Way to extract such information from surveillance videos

Computer vision provides methods for acquiring, processing and analyzing images to be able to extract important information used by artificial systems. [34] Some of these

methods are going to be used to extract, from videos, the parameters mentioned in the section above.

The following section presents some important concepts, as well as the general process for pedestrian and vehicle tracking. It then discusses relevant techniques and the way they should be used.

Most of the tasks mentioned below are developed first on images and can be generalized on videos. Indeed a video is a sequence of images or consecutive frames that are displayed at a frequency called frame rate. The average industry frame rate for video surveillance cameras is equal to 10 frames per seconds [28]. As a comparison, the human eye and its data transmission system can analyze 10-12 images per second. [36] So, if a performant algorithm for pedestrian and vehicle tracking is implemented, the computer should be at least as efficient as a human spotter. However, because computers are not subjected to human factors such as tiredness, they should be more performant than spotters watching a surveillance video.

Object detection

The first step that the algorithm must perform is the detection of objects such as cars, buses, pedestrians, bicycles. The visual content of a video are matrices of pixel intensity for the red, green and blue colors. After a first level of processing, features such as edges, corners, lines or curves can be detected. More processing is done to combine and interpret these features before being able to recognize objects. [37] There is a large variety

of object recognition algorithms but the general concept is the same: datasets containing images with labelled objects are used as pairs of known input-output to train the algorithm.

Main strategies for object detection and selection of the neural network method

The general strategies used for object recognition are the following:

- Appearance-based method: This method uses example images of the object under different lighting and viewing conditions to perform recognition. [38]
- Feature-based Method: This method recognizes features that are compared to a database containing objects characteristics and then recognize objects by using the features extracted. [39]
- Pattern Matching: Small parts of an image that match a template are determined and then the algorithm determines whether the template matches the image or not. [40]
- Artificial Neural Networks: Models that use a set of interconnected units (neurons) having input and outputs and which compute a simple function of their inputs. If x denote the inputs, y its output, a unit (neuron) links the output to the input with the relation: [41]

$$a = w\text{^}Tx + b \text{ and } y = f(a)$$

Where f is the activation function (for instance hyperbolic tangent, sigmoid function) and w are the weights. These weights are tuned during the training process.

Convolutional neural networks were built to improve object recognition on images and have proven to be very efficient. [42]

The most popular object recognition algorithms use the aforementioned methods. Scale-Invariant feature transform (SIFT), Speeded-up robust features (SURF), Principle component analysis (PCA), Linear discriminant analysis (LDA) and Convolutional neural network (CNN) are five promising algorithms used for objects detection. SIFT and SURF are feature-based algorithms, PCA and LDA are appearance-based algorithms and CNN is a neural network algorithm. The performance of these five algorithms is compared in [32] with CNN being the most robust in general and in particular under poor lighting conditions.

Table 2: Comparison of object detection methods and choice of CNN [26]

	SIFT	SURF	PCA	LDA	CNN
Rotation	++	0	+	+	++
Illumination	0	++	-	-	++
Occlusion	++	+	-	-	++

As seen in the table above, convolutional neural networks are the most promising algorithms for object recognition. Some state-of-the art deep-learning frameworks, such as the Caffe framework [33], provide pre-trained convolutional neural networks. Details about the structure of convolutional neural networks are explained below as it helps understand how the Caffe framework is used in this thesis.

Convolutional Neural Network for Object Detection

Convolutional neural networks use arrays of pixel intensities as inputs (figure 6) and provide classes as outputs. These classes in the context of this work are pedestrians, cars, bicycles (it can be many other classes) .

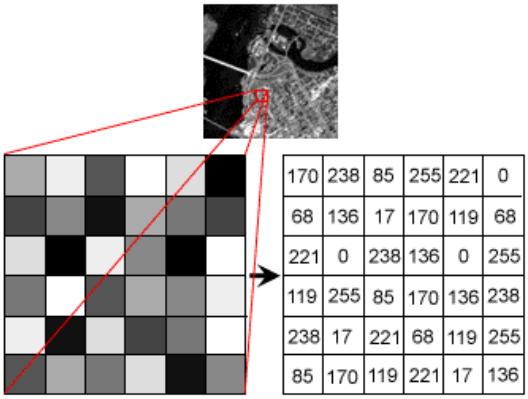


Figure 5: array of pixels as input

The neurons are arranged in three dimensions and are connected to a small portion of the input volume of the layer. [34] This network is called convolutional neural network because it uses convolutional layers that compute dot products between weights of a neuron (or filter) and the small portion they are connected to into the input volume.

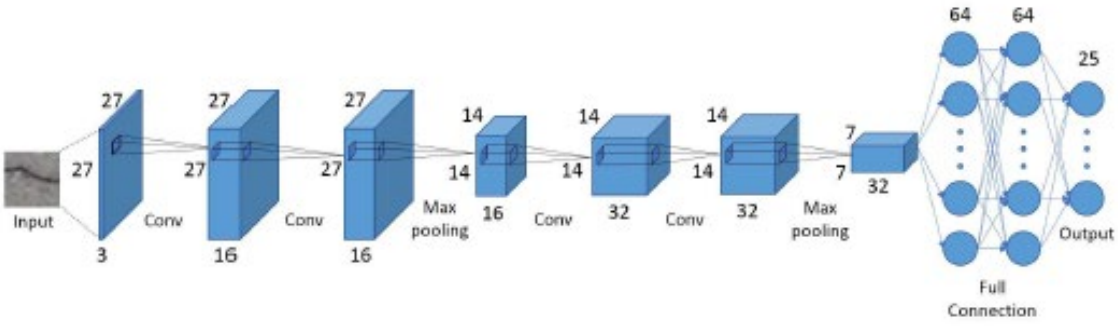


Figure 6: An example of a succession of layers in a convolutional neural network [77]

An example to understand convolutional layers

To better understand the working of a convolutional layer, let's consider an image of 7*7 pixels and a filter, which is an array of 3*3 pixels. This filter is applied to the 7*7 image and it covers a 3*3 area.

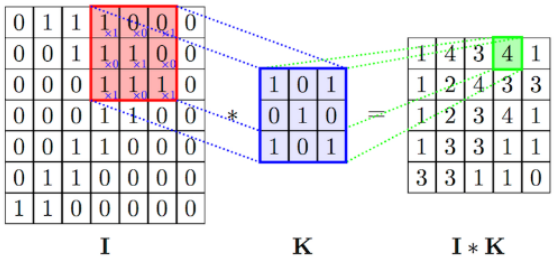


Figure 7: Convolution using a filter K of size 3*3 and an image I of size 7*7

This filter is actually a neuron whose weights are the pixel intensities on the array. On the area covered by the filter, the pixel values of the image are multiplied by the pixel values of the filter (Figure 8). Then the multiplications are summed up, the filter is shifted and the operation with multiplication is done on the new covered area. This process is repeated until the output array is obtained.

This method enables to detect features. Indeed, let's consider the example of a curve (Figure 9), in that case the filter will have high intensity value for pixels corresponding to the curve location. Hence, if the resulting array pixels have low value it means that there is no correspondence with the filter.

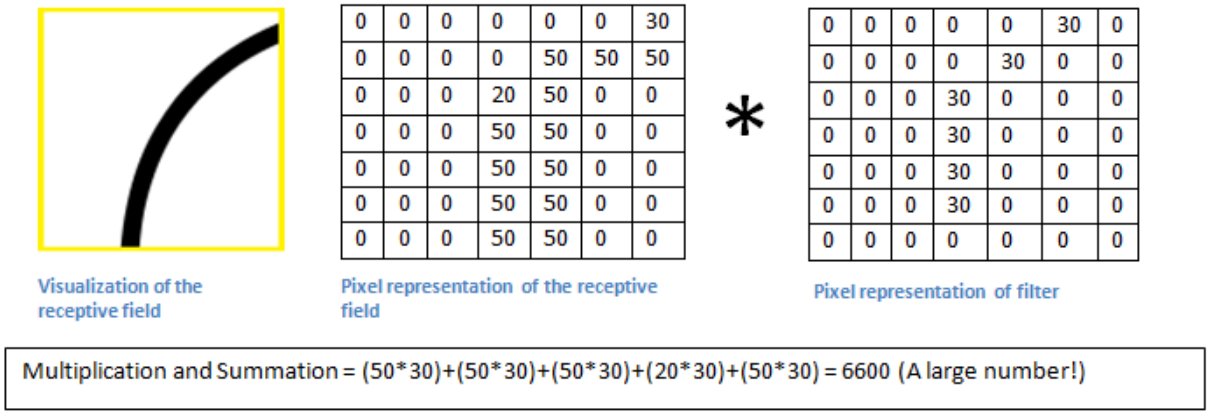
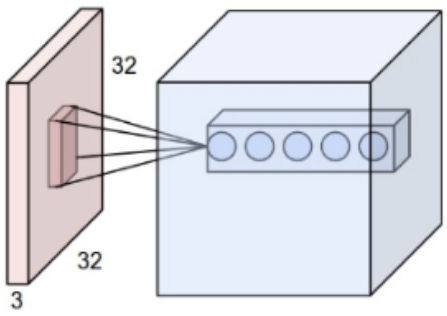


Figure 8: Example of a filter for a curve detection
[65]

Several filters/neurons can be used on the same area to enable the detection of different features. The neurons are located along the depth of the layer, as shown below (Figure 10).

Figure 9: Two layers of a convolutional neural network [34]



The pixel values of a filter are parameters that are updated through the training process. The filter size is an hyper parameter and is fixed before the learning process. The number of filters along the depth of the layer is fixed before the training process too.

Convolutional layers are not the only layers used in a convolutional neural network, but there are the main ones. Pooling layers are a simple example of other layers. They consider cluster of pixel values in a layer and take the maximum value or the average value of this layer to build the pixel value of the following layer. This layer has an important purpose: it controls overfitting, which appears when the model is so tuned to the training examples that it cannot predict well for other sets.

Several objects detection methods exist such as a cascade classifier for vehicles detection, or a haar-like features algorithm for person detection. The convolutional neural network was selected for the reasons mentioned above.

Caffe framework for object recognition

The neural network used in this thesis is provided by the Caffe framework. Caffe is a deep-learning framework that provides fast classification tools for object and speech recognition. [33] This framework was selected because it is expected to provide fast object detection capabilities when implemented on the GTPD surveillance videos.

The Caffe framework for object detection provides two files: a protext file and a Caffe model file. The protext file is similar to a JSON file and contains the network architecture. The succession of layers and their hyperparameters, such as the number of neurons along the depth, or the neuron size and shift step, are stored into this file. The Caffe model contains the weights that are determined during the training. In this thesis, the Caffe

framework is used to detect cars, buses, bicycles and pedestrians on videos. Before using these two files the video must be read by the algorithm and transform into an object called "blob" that is the standard array for the Caffe framework and is used as input of the neural network. After using the Caffe files the location of pedestrians and vehicles must be determined.

Pedestrians, vehicles and buses tracking

Once detection has been performed, it is possible to identify objects and determine their locations on the video, however there is no identity associated with these objects. It means that the algorithm does not build a relation between a pedestrian A in the frame i and the same pedestrian A in the frame i+1. It thinks that both pedestrians are different. A matching process between successive frames must be performed to follow pedestrians and vehicles within different frames. During the tracking process, object detection is performed at a frequency n to detect new incoming objects within the field of view and to help and verify tracking. [37] Between the frame 1 and the frame n, objects are tracked, namely target regions in successive frames are matched to follow the object. This analysis task can be done using OpenCV and a new library dlib in order to determine trajectories and then compute speed for pedestrians and vehicles.

Traffic light color detection and sidewalk location

OpenCV provides libraries that help to detect colors and thus enable to determine the traffic light color. The sidewalk location can be obtained by manually tuning its position on the first frame of the video.

Consequently, thanks to computer vision methods and tools, the following parameters can be extracted from videos: vehicles and pedestrians path trajectories, traffic light color and sidewalk location. Then, parameters such as speeds, location and vehicle-pedestrian distance, vehicle distance to the crosswalk and pedestrian distance to the road can be compared to security thresholds to be classified as violations or dangerous situations. As an example, the vehicle speed is compared to the speed limit and then the situation is classified into vehicle speeding or not.

This lead to Hypothesis 1:

If computer vision techniques are applied on traffic camera data then high-risk situations can be automatically detected.

These high-risk situations are then recorded into a database. The developed tool is designed to be used in real time for all surveillance video cameras located at intersections on campus. Hence, a huge amount of information need to be recorded and a huge amount of computation needs to be performed due to the high number of parameters to be computed for each pedestrian and vehicle. Consequently, a good way to enable a fast computation is to install the code on a server. Moreover, the server can be used to store the database which contains the conflict data. These servers do not need to be located on site and consequently do not need to be acquired by GTPD, indeed cloud computing enables to use servers located somewhere else. However, the storage price of the database on a server increases with the storage size needed. It is consequently important to select the high-risk pedestrian-vehicle encounters that are going to be stored in the database. The difficulty of the task is high

because high risk situations have a different nature and are consequently hard to compare. For instance, is speeding more risky than a low distance pedestrian-vehicle situation? How to determine the high risk situations that are going to be stored in the database?

This lead to the following sub-question,

RQ1.1: How can high-risk situations be detected and categorized?

3.1.2.3 Second set of information required to enable the desired capabilities

First, there is a need to assess the severity of all high risk-situations to determine whether they shall be recorded into the database or not. Secondly, there is a need to classify these situations: is it speeding? Is it jaywalking?

To address the first requirement, indicators for the severity of conflicts must be computed. As defined previously, a conflict is an observable situation in which two or more road users approach each other in time and space such that there is risk of collision if their trajectory and speed remain unchanged. [10]

A method to evaluate the severity of conflicts could consist in computing two indicators [35]: Pdirection, which provides an information about the pedestrian's motion direction compared to the road direction, and Pposition, which provides information about the location of the pedestrian compared to the road. The level of risk to pedestrians is evaluated using a combination of Pposition and Pdirection. The method described in [35] does not account for vehicles. Indicators Pposition2 which compares the proximity of vehicles to crosswalk and Pdirection2 which provides extra space information about the

driver's path compared to the pedestrian's one could be computed to complete these methods. However, no information about vehicles speed is considered, which makes this method inadequate to answer the first need.

In the literature, assessing the level of risk in vehicle-pedestrian interactions often requires the computation of surrogate safety measures that measure both the spatial and temporal proximity of road-users. [29] These surrogate safety measures are indicators for the severity of near-accidents. The most commonly used are the following [30]:

- Time to collision TTC: Time until two road users collide if they remain at their present speed and on the same trajectory.
- Gap Time: Time difference between the second user arriving at the conflict point and the first user leaving it when both continue with same speed and trajectory. It is an indicator of the road user closeness.
- Deceleration to Safety (DST): The deceleration required so that crossing vehicles avoid a collision.
- Post encroachment time (PET): Time between the first road user leaves a common zone and the second user actually arrives in that zone.

The minimum values of TTC and GT and the maximum value of DST and PET are indicators of the risk of a conflict.

Vehicle-pedestrian encounters can be classified into three patterns [38]:

- Pattern 1: TTC and GT reach their minimum in the middle of the interactive process. It occurs when the driver or the pedestrian react significantly such as to reduce his speed to avoid a collision. This pattern is called hard interaction.

- Pattern 2: TTC and GT decrease in the same way and reach their minimum at the end of the interactive process. The pedestrian and the driver kept their speed almost constant during the process. This pattern is called no interaction.

- Pattern 3: TTC and GT follow different trend through the interactive process, the driver adjusts its speed. This pattern is called soft interaction.

Table 3: Evolution of TTC and GT and corresponding patterns

Pattern 1	Pattern 2	Pattern3
TTC and GT reach their minimum in the middle of the interaction. Significant reaction of the driver or the pedestrian such as obvious speed reduction	TTC and GT reach their minimum at the end of the process. Pedestrian and driver kept their speed constants	TTC and GT follow different trend. For instance when the driver adjusts its speed.

Once the pattern is determined, the situation is evaluated as dangerous, critical or safe using specific coefficients for a given pattern. The indicator required to evaluate the risk in pattern 1 is TTC, for pattern 2 is PET and for pattern 3 is TTC and PET combined. Pattern 1 is a safe interaction if TTC>3.0s, it is a critical one if 1.5<TTC<3.0s and a dangerous one if TTC<1.5s. Pattern 2 is a safe interaction if PET>3.0s, it is critical if 1.0s<PET<3.0s and a conflict if PET<1.0s. Pattern 3 is a safe interaction if TTC>3.0s and

PET>3.0s, it is a conflict situation if TTC<1.5s and PET<1.0s and a critical interaction otherwise. [47]

This method enables a robust evaluation of the criticality of complex pedestrian-vehicle interactions. That's why it should be implemented as part of this research in order to select high-risk situations among the pedestrian-vehicle interactions recorded by videos.

This lead to the following hypothesis:

Hypothesis 1.1: If TTC, GT and PET are used then high-risk situations can be defined

It is expected that, using TTC, GT and PET, the severity of conflicts of different natures can be assessed. Indicators can be used for all complex pedestrian-vehicle interactions. However, these indicators do not classify these interactions. To help the Police Department to have a better understanding of hazardous behaviors at intersections it is necessary to determine whether the situation is linked to speeding, jaywalking, red-light running or late yielding. Hence, it is necessary to get the nature of the interaction and parameters that describe it (for example the vehicle speed for speeding).

To address this second requirement, parameters such as velocity of pedestrians and vehicles, minimum distance between them and deceleration during braking must be computed. [26] A way to classify these situations is to compare these parameters to thresholds. For example, the speed is compared to the speed limit in the area to classify the situation as speeding or not into the database. Once the vehicle location is obtained, the distance vehicle-crosswalk is computed and compared to a safety distance that the vehicle

must respect. To determine whether the situation is a jaywalking or not, the pedestrian location is compared to the roadway location and the traffic light color at that moment. If the pedestrian is on the roadway when the walking signal is red, then the trajectory is analyzed to know whether the pedestrian started to cross the street before the light turned red or not.

By implementing rules such as the ones described above it may be possible to classify the situations as jaywalking, speeding etc. into the database.

This lead to the following hypothesis:

Hypothesis 1.2: If reference thresholds are applied on speed, position, pedestrian-vehicle distance and deceleration during braking then traffic situations can be identified and classified.

The interactions that are classified belong to one of these situations: vehicles yielding too late to pedestrians at crosswalks, speeding by vehicles at crosswalks, vehicles stopping too close to crosswalks, pedestrians stepping into traffic, pedestrian or vehicle crossing against the traffic light signal. The surrogate safety measures enable to account for driver-driver situations and enable to divide the incidents into dangerous, safe or critical. Consequently, both methods are necessary: use of TTC, GT and PET and comparison of speed and location to thresholds.

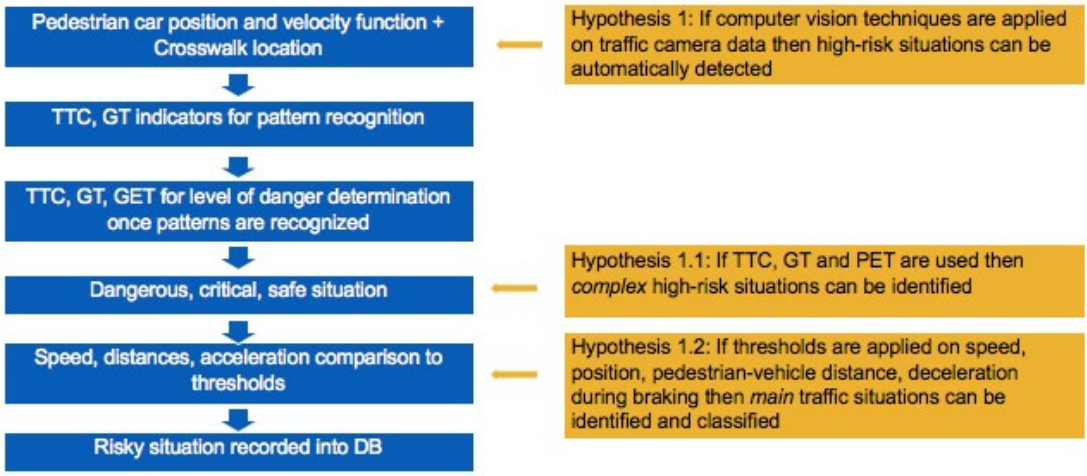


Figure 10: Steps to detect and classify high risk situations into a database

3.1.2.4 Summary

Machine learning methods for object detection and computer vision techniques for object tracking enable the extraction of parameters such as vehicle or pedestrian speed and location. These parameters are required to detect and analyze high risk situations for pedestrians using surveillance video. Indicators are computed using these parameters to assess the dangerousness of a situation, only critical and dangerous situations are considered. To sort the information extracted into the database and to get statistics about high risk situations for pedestrians, the situations must be classified into jaywalking, red-light running, speeding, late yielding. To classify these situations, speed, location and light color are extracted from videos and compare to threshold values.

3.2 Optimization of traffic light control

Once detailed data about high risk situations at intersections is extracted from videos, enforcement operations can be automatically implemented and unlawful behavior

can be penalized automatically. Moreover, the Police Department can use this data to determine countermeasures needed, such as implementing a new traffic light control system to avoid jaywalking risks.

A large portion of pedestrian-vehicle collisions occur at intersections. Vehicles are not the only ones responsible for these accidents, as pedestrian behavior can result in accidents or injuries to occur. Indeed, pedestrians are inclined to ignore traffic signal indicators when delays are significant [48]. Moreover, the campus infrastructure size is static, urban road network expansion is limited but the population increases on campus. Vehicle and pedestrian traffic flow has increased at intersections, leading to additional risk for traffic congestion and pedestrian-vehicle conflicts.

Adapting traffic light control to vehicle and pedestrian flow is a way to avoid jaywalking and adapt traffic signal to higher traffic volumes. On campus, most traffic lights are regulated by pre-determined green times that can be slightly modified by pedestrian call buttons. Currently, campus traffic lights are not optimal at the intersection between 5th Street NW and Spring Street NW, for example. The vehicles and pedestrians waiting time is not adapted to rush hours, hence the risk of jaywalking and delay for road users are high at this intersection. Consequently, it is necessary to optimize the waiting times of both vehicles and pedestrians at this intersection.

This lead to the research question 2,

How to optimize the traffic light control to reduce the waiting time and consequently the risk of jaywalking?

Instructions that use real time traffic flow information and that consider pedestrians waiting time must be implemented at this crossing. Moreover, this traffic light control should be adapted to rare events such as football games.

In a first section are presented the current traffic light control systems and in a second section are presented two promising methods to optimize traffic light control.

3.2.1 Benchmark of traffic light control systems

The most common way to control traffic light uses fixed-time methods. Surveys of traffic flow at different periods in the year and time of the day are done to determine green time for each intersection's traffic light. [41] Hence, a large amount of data is required to implement this method. The traffic light color changes can be pre-configured to different values based on different times of the day. Some of them are coordinated so that vehicles meet a continuous series of green lights. However, this method is not efficient for traffic condition different from the ones prevailing during the survey. Additionally, it requires a very large amount of traffic data which is often manually collected, meaning its implementation is laborious.

Another strategy called adaptive traffic light uses detectors to recognize current traffic amount at an intersection and extend or terminate green time accordingly. [41] Sensors or surveillance cameras are used to detect whether vehicles are present or not and the traffic light color is adapted based on this information. Adaptive traffic light control systems enable to distribute green light equitably, reduce congestion because they create smoother flow and improve travel time. Excess delay increases labor cost and decreases productivity for businesses. By reducing travel time, adaptive methods reduce costs due to

excess delay and fuel consumption, leading to major improvement when compared to fixed-time methods. On average, they improve travel time by more than 10 percent. [42] In spite of their capability to reduce delays, adaptive signal control technologies have been implemented on less than 1 percent of traffic signals. [42]

SCOOT (Split Cycle Offset Optimization Technique) is an adaptive traffic light system that minimizes the sum of average queue in a specific area [51]. SCATS (Sydney Coordinated Adaptive Traffic System) aims to maximize the throughput by choosing the appropriate signal timing from a library [52]. However, these two methods are inefficient for saturated traffic. OPAC (Optimization Policies for Adaptive Control) and RHODES (Real Time Hierarchical Optimized Distributed Effective System) employ dynamic traffic models and traffic measurements to minimize the total intersections delay but these algorithms have an exponential complexity that make them inefficient for a large scale network [53]. Finally, the method TUC (Traffic-responsive Urban Control) requires simplified measurements compared to the previous methods and is efficient for saturated traffic conditions, however its strategy needs to be redesigned if the network is modified or expended. [54] This strategies are summarized in Table 4.

Table 4: Summary of the main adaptive traffic light methods

Methods	Working	Drawbacks
SCOOT [51]	Minimize the sum of the average queues	Inefficient for saturated traffic conditions
SCATS [52]	Maximize the throughput by using a signal timing from a library	Inefficient for saturated traffic conditions
OPAC and RHODES [53]	Minimize the total intersection delay	Exponential complexity

TUC [54]	Minimize delay for saturated traffic conditions	Need to be redesigned if the network is expanded

Learning-based adaptive signal control systems have been developed and outperformed the existing methods. Moreover they do not require a model of the environment to be implemented and consequently no assumption concerning the environment must be made. They use machine learning techniques to optimize the traffic light control and enable real-time instructions. [55]

3.2.2 Two promising methods for traffic light control

Neural Network and Reinforcement learning are two promising methods for learning-based adaptive signal control. [55] Before giving more details about these two methods, machine learning and especially reinforcement learning are briefly presented.

3.2.2.1 Machine Learning: supervised, unsupervised and reinforcement learning

Machine learning is the field of artificial intelligence which provides machines the with the ability to learn. Learning implies being able to reuse in the future what has been learnt in the past. [41] The machine changes its structure or program by learning from data and operates these changes in a way to improve its performance or its main task. [44]

There are three main types of machine learning techniques:

-Supervised learning where the machine uses pairs of known input-output to fit the model.

-Unsupervised learning. This method does not have labelled outputs and attempts to learn the structure of the data by using lots of example of an object.

-Reinforcement learning, where an agent learns the best decision-making policy. More details about reinforcement learning are given below as it is one of the strategy selected for traffic light control optimization.

Reinforcement Learning

Reinforcement learning is the state-of-the art technique used to determine strategies (in games for example). This method trains the model using a system of rewards and regrets. [41]

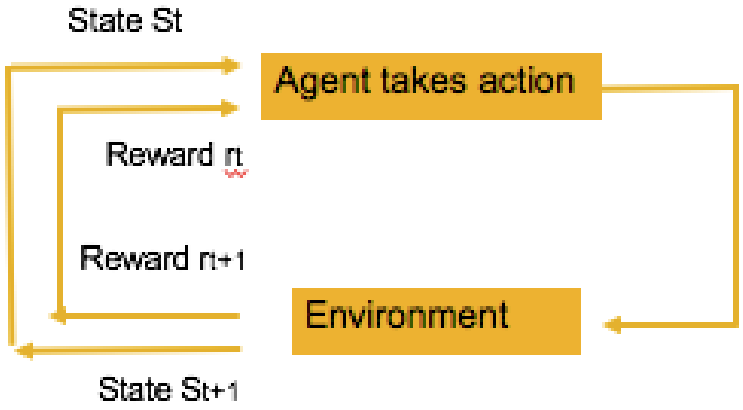


Figure 11: Reinforcement learning process [58]

An agent performs an action that is chosen from a number of possible actions to get a good reward. The result produced by this action on the environment is evaluated and the reward is determined based on this evaluation. The algorithm attempts to learn a sequence of actions that will maximize the long term reward which is built using short term rewards from previous states. The long-term reward is learnt when an agent interacts with an environment through many trials and errors. This is the cost function that is optimized

during the process. The long term reward is called Q-function in the case of Q-learning which is the most common form of Reinforcement learning. The Q-function depends on the reward and the previous Q-learning functions. The update rule for the Q-function is given by the following equation where lr is the learning rate, dr the discount rate, s the state, a the action and s' the new state obtained by combining the action a and the environment. [55]

$$Q(s,a) = (1 - lr)Q(s,a) + lr(reward + dr * \max_{a} Q(s',a))$$

During the reinforcement learning process, a table which contains all value of the Q function for state-action pairs is built and used for the computation of the next Q-value function. The Q-value table is a representation of the long-term reward an agent would receive when taking this action at this particular state. An example of such a table is presented in figure 14.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Figure 12: Q-table

Reinforcement learning consists in learning these Q-values while taking actions. The Q-table is used to determine the next action. A common strategy for the choice of the next action is to use the ε-greedy exploration. During the ε-greedy strategy, the maximum Q-value in the Q-table and the corresponding action are selected. However sometimes a

random action is selected instead of the optimal one. Indeed, it is necessary to explore others action values to be sure to explore all possible options and to get the best option available in the long term. The ε-greedy strategy allows to select a random action with a probability ε and the action which maximizes the reward with a probability 1-ε.

3.2.2.2 Neural Network and Reinforcement learning strategies for traffic light control

Some terms must be defined to introduce the working of Neural Network and Reinforcement learning algorithms for traffic light control:

- Phase [45]: controller timing unit associated with the control of one or more movements. Example: the lights for pedestrians being green and the other lights being red is a phase.
- Cycle [45]: Total time to complete one sequence of signalization for all movements at an intersection. The cycle is split into a sequence of green signals.
- Interval [45]: Duration of time during which the signal indications do not change and is set to red, orange or green.

Neural Network strategy for traffic light optimization

To control traffic lights at an intersection with four phases, an effective method consists in using a neural network with four input neurons, ten neurons in the hidden layer and four neurons in the output layer [55]. The inputs are the lengths of queues, and the outputs are the NN estimations of green time for each phase for a cycle.

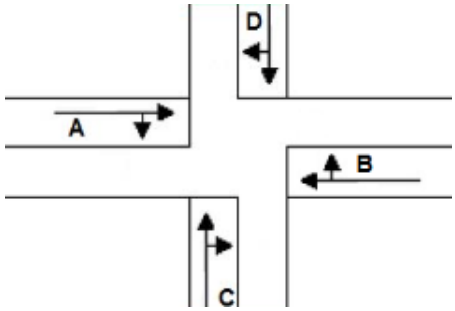


Figure 13: Intersection considered in [46]

The green light times computed by the Neural Network are then used as inputs in a simulation environment. The queue lengths and the average delay are determined through the simulation, which is performed during several cycles. The queue lengths are the new inputs of the Neural Network algorithm. This logic is presented in Figure 16.

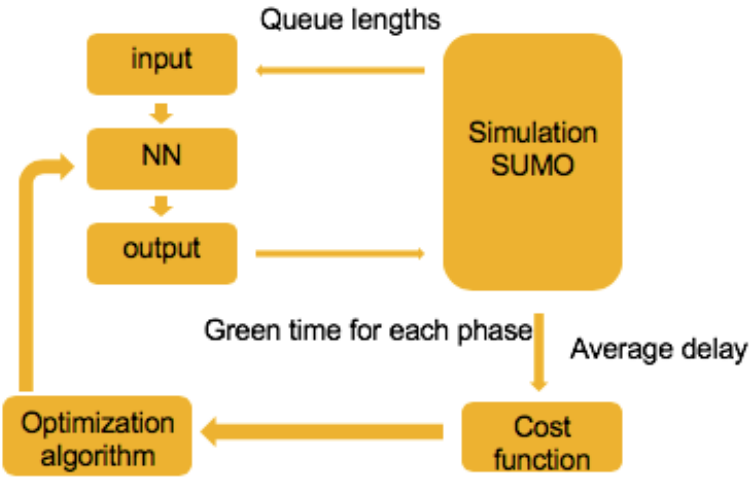


Figure 14: Integration of the neural network algorithm into a simulation environment. [55]

The average delay time over each complete simulation is the cost function that is used to optimize the weighs of the Neural Network. The minimization of the cost function by an optimization algorithm leads to a weight update for the Neural Network. This process is done until there is no further improvement of the total delay during several iterations.

The method presented above does not account for pedestrian waiting time so it is not adapted to the intersection between 5th street NW and Spring street NW, where the pedestrian waiting time is high during lunch time and the jaywalking risk is consequently important. Including the pedestrian waiting time into the total delay and the pedestrian phase into the cycle could lead to a more adapted traffic light control strategy. This lead to the following hypothesis:

Hypothesis 2: If pedestrian activity is included into the Neural Network model then a more adapted traffic light control can be obtained.

The cost function that enable to update the neural network weights will depend on vehicles and pedestrians waiting time. Weights can be used to appropriately balance the impact of pedestrians waiting time compared to vehicles waiting time in this cost function. This lead to the following hypothesis:

Hypotheses 3: If weights are allowed to vary in the neural network models then pedestrians behavior can be appropriately accounted for.

Reinforcement learning strategy for the traffic light optimization

An alternative method to neural network consists in using reinforcement learning. When we deal with Reinforcement learning, three parameters need to be defined

- State which describes the current situation. For the traffic light control problem, the state can be the length of waiting queues for pedestrians and vehicles for all directions.
- Action: what an agent can do in each state, which leads to a new state. In the case of traffic light control, this corresponds to the timing change of traffic light.
- Reward, which is an evaluation of the action in the environment. The reward is a function of weights for waiting times and of the pedestrian and vehicular queue lengths.

During the reinforcement learning process, an action is performed by an agent and its impact on the environment is evaluated to determine the reward. A simulation environment is used to train the reinforcement learning model. The traffic state (delays and

green times for one cycle) and a map of the intersection are the inputs of the simulation that is run in parallel to the optimization algorithm. The output is the new state: the new delays. These new delays are then used to compute the reward.

Most traffic light control methods using reinforcement learning do not consider pedestrians. Because reinforcement learning methods have proven to be efficient to manage motorized traffic [55], we can assume that integrating pedestrians into a Reinforcement learning model will lead to a better traffic light control for both motorized and non-motorized traffic. This lead to the following hypothesis:

Hypothesis 4: If pedestrians are included into the reinforcement learning model then a more optimized traffic light control can be obtained.

Ying Liu, Lei Liu and al worked on a multi-agent Q learning strategy [41], which considers non-motorized vehicles. Their work includes neighboring intersections into the model, however no traffic data at neighboring intersections are provided for this research. Hence, the method of Ying Liu et al must be adapted for a single intersection.

In the work of Ying Liu, lei Liu et al, the state is the length of pedestrian waiting queues at the specific intersection and of vehicles for all directions at all intersections in the neighborhood. An action is a timing change of a traffic light and the immediate reward is a function of waiting times for vehicles at the intersection and at other intersections, of pedestrians waiting time at the intersection and of weights for each waiting times. This work considers the vehicles' waiting time at neighboring intersections but do not consider

the pedestrians waiting time at neighboring intersections. This translated into pedestrians being penalized compared to vehicles. If the weights in the immediate reward are modified to include pedestrians at the same level as all other vehicles, then the method can lead to a more optimized traffic light control system at the crossing between 5th Street NW and Spring Street NW.

The first reinforcement learning method presented above gave more importance to vehicles than pedestrians because it considers vehicles at neighboring intersections but it does not consider pedestrians there. A promising reinforcement learning method that focus on a unique intersection without pedestrian is implemented in [47]. This method is adapted to normal traffic flow rate, rush hours and rare events such as football games. Moreover, the learning speed of this method outperforms a large number of other reinforcement learning strategies applied to traffic light control. This method divides the space into a grid such as only one vehicle can fit in a grid. A matrix is built such as $M_{ij1} = 1$ if there is a vehicle in the grid (i,j) and 0 otherwise. $M_{ij2} = speed\ of\ the\ car$ if a vehicle is present in the grid(i,j). The state of the reinforcement learning process is given by this matrix (M_{ijk}).

An action is defined as the timing change of a traffic light. The succession of actions is fixed in this method. For the intersection between 5th Street NW and Spring Street NW the succession of actions is the following: the light is green for A and B, then it is green for C and then the light is green for all pedestrians D.

The reward is the waiting time difference between the current state and the new state once the action is taken.

$$Reward = W_t - W_{t+1}$$

Where W_t is the sum of waiting time of each vehicle. However, this method does not take pedestrians into account. Including pedestrians waiting time into the reward by using a weight factor and modifying the grids at sidewalks to include pedestrians into the state would help to integrate pedestrians into the model. Thus, the following hypothesis can be formulated:

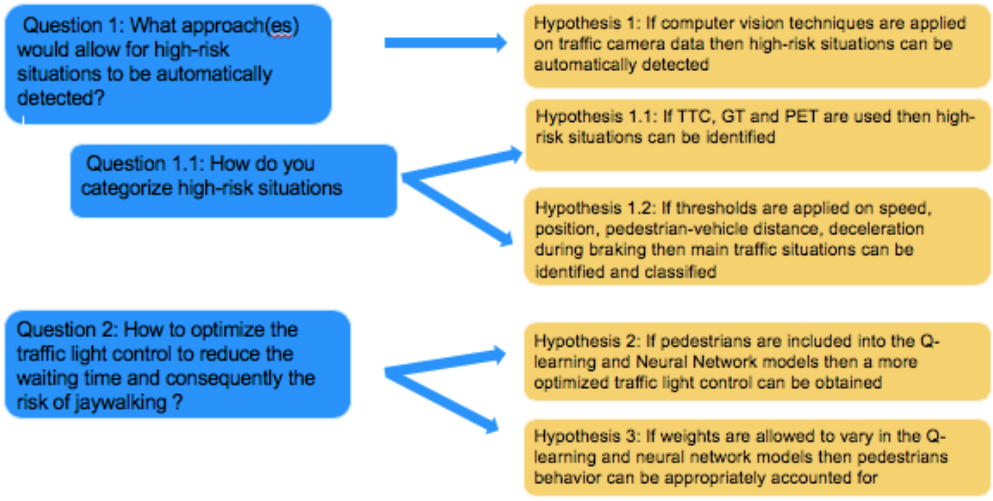
Hypothesis 5: If weights are allowed to vary in the Q-learning reward then pedestrians behavior can be appropriately accounted for.

3.3 Mapping of Research Questions and Hypotheses

Below is a diagram of the research questions and the hypotheses that need to be validated to answer the main research question of this thesis. Answering Question 1 and 1.1 should lead to a method that collect and classify risky-situations in an automated way. The answers to question 2 should lead to an optimized traffic light control system that account for pedestrians.

Some hypotheses were combined. Indeed the hypotheses 2 and 3 presented in the previous sub-section are both related to the same need which is to include pedestrians into the traffic light optimization process. Hence they are gathered into a same hypothesis: Hypothesis 2 on the diagram. Hypotheses 4 and 5 in the previous sub-section refer to the weights modification into the optimization methods and are gathered into the same Hypothesis 3 on the diagram.

Figure 15: Mapping of Research Questions and Hypotheses



CHAPTER 4. TECHNICAL APPROACH

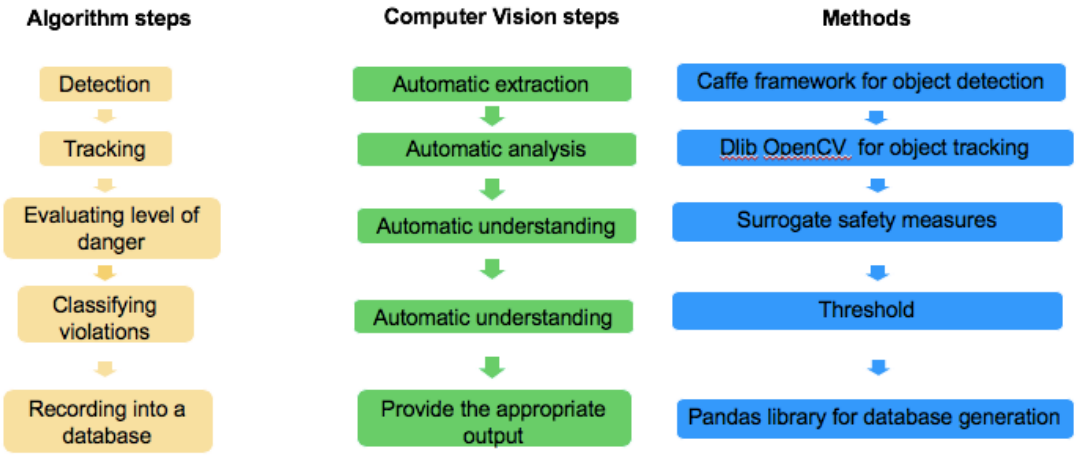
4.1 General approach

The use of computer vision on surveillance videos located at intersections will help to obtain data about conflicts at the intersections. At first, parameters such as speed and trajectory for pedestrians and vehicles must be extracted from videos. Computer vision provides techniques that enable to detect and track objects and then trajectories and speeds can be determined. These steps are done using the state-of-the-art computer vision and image processing framework OpenCV.

Once these parameters are obtained, conflicts can be identified and their severity determined using surrogate safety indicators such as Time-To-Collision or Gap Time. All complex pedestrian-vehicle interactions can be identified and evaluated using these indicators. Then, metrics such as speed and location must be compared to thresholds to classify these situations as jaywalking, speeding etc. At the end of this step a database which contains high-risk situations and details about it is built.

This tables enable to perform enforcement automatically and to determine the needs for countermeasures. The diagram below describe the process mentioned previously.

Figure 16: Computer vision steps to build the tool that automatically detect conflicts



After executing these steps, primary risks to pedestrians can be identified at each intersection and measures can be determined by the police department to reduce the risks. Moreover, automated enforcement can be implemented in order to improve driver behavior at these intersections.

A current problem is the traffic light scheduling at the intersection between 5th street NW and Spring street NW. Indeed the pedestrian waiting time during lunch time is high at this intersection and consequently pedestrians are inclined to ignore the traffic light signal. Moreover, the vehicle waiting time in 5th Street NW is high during rush hours which leads to traffic congestions. Machine learning techniques should enable to optimize traffic light control for an isolated intersection with pedestrians. A neural network strategy and two reinforcement learning strategies should be implemented and compared to baseline methods for traffic light control.

The general approach is illustrated in the figure below.

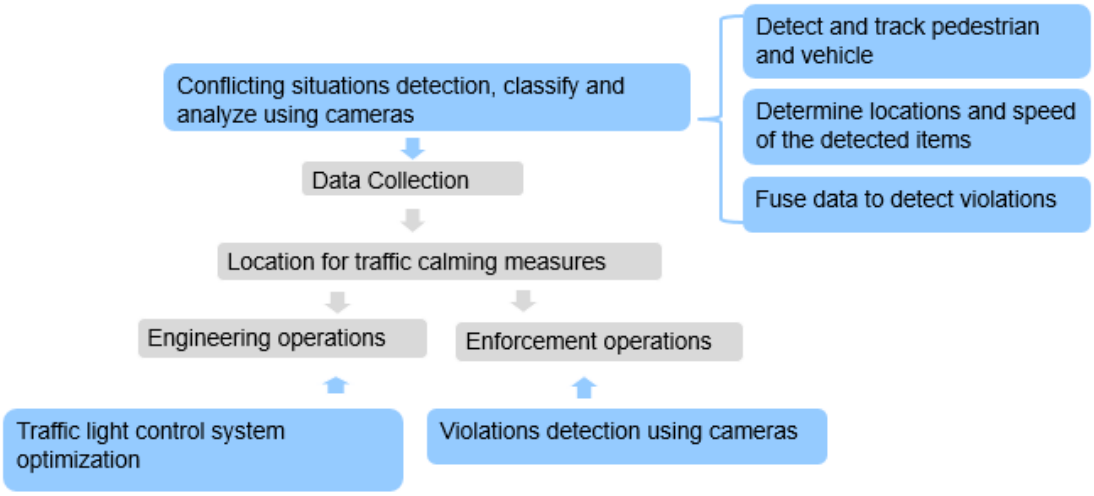


Figure 17: General Approach

4.2 Approach to automatically detect high-risk situations

4.2.1 Computer vision to get trajectories, speeds and traffic light color

The first step leverage computer vision to extract trajectories and speed from cameras and be able to determine a violation or a high risk situation. Detection of violations

Once trajectories, speeds and traffic light color are extracted from videos, violations can be detected. The previous data concerning vehicles, pedestrians and traffic light can be stored into tables. Then comparing traffic light color and vehicle location or walking signal color and pedestrian location help determine red light running and jaywalking. Comparing vehicle speeds to speed limits enable to determine speeding in an automated way.

Once the violations are detected, the frequency of red light running, jaywalking and speeding can be determined and could be used to complete the Police record database.

4.2.2 Selection of high risk situations only

The final database cannot contain all information because it increases storage needs and costs. Consequently it is important to select high-risk interactions only. Three surrogate safety indicators: TTC, GT and PET can be computed using information extracted through computer vision. In the literature, these indicators help to determine the severity of complex vehicle-pedestrian interactions. Computing these three factors for interactions at campus intersections should help to select high-risk situations that will be recorded into the database.

A comparison with thresholds used for Hypothesis 1.2 will be performed to evaluate whether a situation selected through the use of surrogate safety indicators represents really a high-risk situation. Metrics such as speed, pedestrian-vehicle distance, deceleration during braking will be computed for the pre-selected situation. These metrics will be compared to thresholds to determine whether there is really a violation or a high risk-situation. If both steps (use of surrogate safety indicators and use of thresholds) are relevant, then Hypothesis 1.1. will be validated, otherwise Hypothesis 1.1 will be rejected.

4.2.3 Comparison of the database built to the Police report database

First, interactions are selected using TTC, GT and PET and then the selection is confirmed by the computation of speed or pedestrian-vehicle distance.

Then a database containing high risk situations can be built. Statistics about crashes and injuries can be determined using this database. The data extracted should be compared to the data obtained through the police report database to check whether the information

obtained is relevant. This data can also be used to complete the current police record database. This comparison will enable to validate or reject Hypothesis 1.2.

4.3 Approach to optimize traffic light control

4.3.1 Pedestrians integration

Neural network and reinforcement learning can be adapted to a single intersection for pedestrian problem. It can be done by integrating the pedestrians waiting time into the state and into the reward for the reinforcement learning strategy and by integrating the pedestrians waiting time into the input neurons and the pedestrians green time into the output neurons for the neural network method.

For the specific case of the intersection between 5th street NW and Spring street NW (Figure below), there are currently three phases: the light is green for A and B, or the light is green for C or the light is green for all pedestrians.

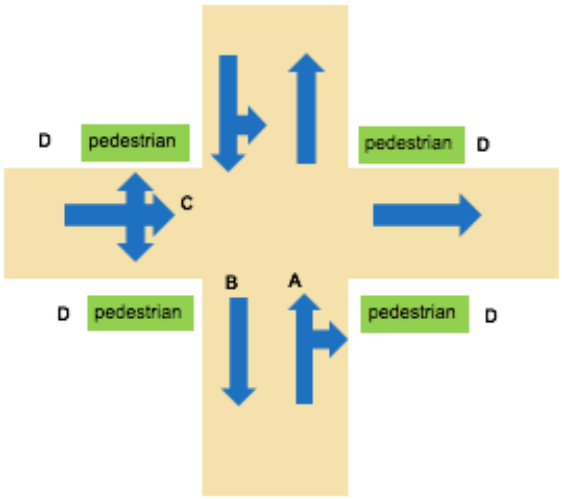


Figure 18: 5th Street NW/Spring Street NW intersection

The inputs (queues lengths) and outputs (green times) of the neural network can be described as follows. There are three vehicle queue lengths A, B and C and one pedestrian queue length D, which is the sum of the four queue lengths at each corner. There are three green time phases: when the light is green for A and B, when the light is green for C and when the light is green for D: the pedestrians lights are green in the same time for all four corners. Consequently, the neural network should have four inputs layer for the four queues lengths and 3 outputs layers for the green times. The number of hidden layer is an hyperparameter (a parameter which is fixed before the training) and can be adapted during the training. In this work the number of hidden layers will be initialized to ten.

The resulting pedestrians and vehicles waiting time will be computed to determine if the new methods lead to an improvement. If operating these changes lead to a reduction in pedestrians waiting time without penalizing too much the vehicles waiting time, then a

change in the weights in the Q-learning and neural network models appropriately account for pedestrians and hypothesis 3 can be validated. The validation process is described in the figure below.

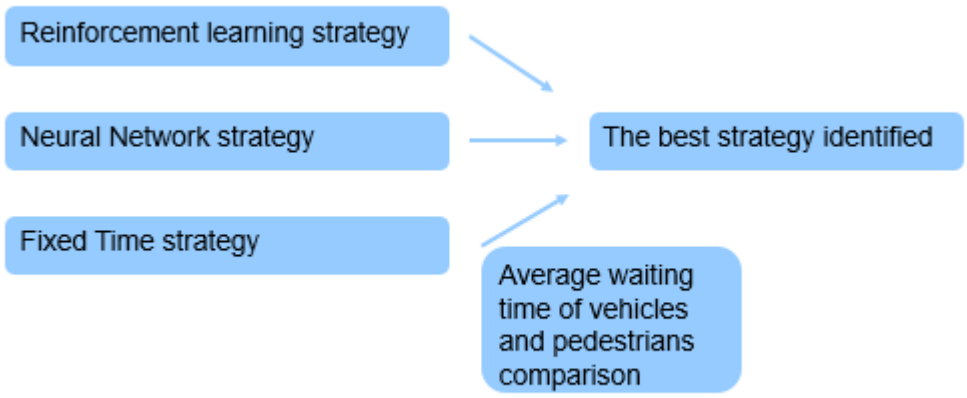


Figure 19: Validation process for Hypothesis 3

4.3.2 Comparison to baselines methods

To evaluate the hypothesis 2 the new neural network model and the two reinforcement learning methods adapted to a single intersection with pedestrians must be compared to a baseline method. A common strategy is to compare the optimized method to a fixed-time method. If the pedestrians and vehicles waiting times decrease when Neural Network or Reinforcement learning strategies are used, then these methods lead to a more optimized strategy and hypothesis 2 would be validated.

The work could stop here, however it is relevant to prove that once pedestrians are included into neural network and reinforcement learning models then the strategies perform

better than common adaptive strategies that take pedestrians into account (fixed-time control systems don't consider pedestrians waiting time).

Common adaptive traffic signal methods require a model of environment to describe pedestrians and vehicles flow. Multiple papers in literature refers to the Akcelik equations to estimate delays at traffic signals. [59]

In [49] a recent adaptive method that minimizes the trade-off delays between motorized and non-motorized traffic and that use delays model is presented. This model can be used as reference model to evaluate the reinforcement learning and neural network strategies. First, a pedestrian model is built, then a vehicle model is built and finally both models are integrated together. The pedestrian model is built by using stage constraints (for instance, for the case of the intersection between 5th street NW and Spring street NW the four pedestrian lights are green at the same time), crosswalk capacity constraint and Akcelic equations for delay. Then, the cost function to minimize the pedestrian delay is determined using this information. In the same way the cost function for the vehicle delay is determined. Once both models have been built they are combined together using weights to balance pedestrians and vehicles delay and it leads to a model that considers pedestrian and vehicle.

The chart below describe the steps to follow to build this traffic light control system that can be used as a benchmark method.

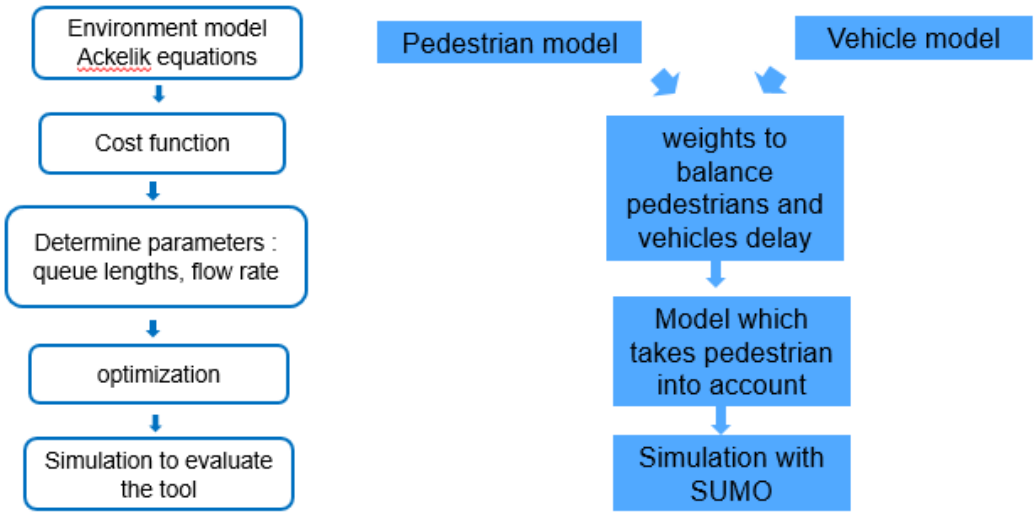


Figure 20: Steps to build the benchmark traffic light control system

Implementing this last method and comparing the total delay to the one of neural network and reinforcement learning strategies will help answer Hypothesis 2. If reinforcement learning and neural network perform better that the third adaptive method, then they lead to a more optimized traffic light system.

4.3.3 Use of a simulation environment for the comparison

For the comparison and the tuning of the model, a simulation tool will be used.

As seen previously, the Neural Network strategy provides green time for each phase. In order to optimize its weight a cost function is computed using the total delay which is determined through a simulation. The simulation environment takes the map of the intersection surrounding, the green times and current delays as input and returns a new

state of the intersection. New waiting times are obtained through the simulation and the cost function can be computed.

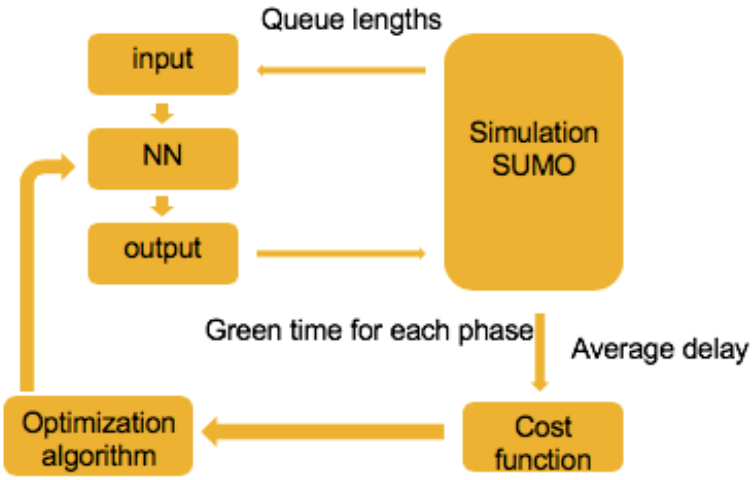


Figure 21: The optimization algorithm must run in parallel to a simulation environment

The reinforcement learning requires a simulation environment too as it needs to evaluate the impact of an action on the environment.

Finally, a simulation environment is required to compare the final models. In particular, similar traffic conditions will be implemented into the simulation environment and each strategy will provide a new state of queue lengths. Then the total waiting time will be compared for the different models to determine the most performant one.

The simulation environment SUMO will be used as it takes pedestrian flows into account and provides an accurate environment model and vehicle traffic model. [61]

The next chapter discusses the automatic detection, classification and storage of violations and hazardous situations at intersections.

CHAPTER 5. AUTOMOATIC DETECTION OF CONFLICT SITUATIONS AT INTERSECTIONS

5.1 General approach

In a first time pedestrians and vehicles are detected on videos. Once these items are detected a tracking stage starts during a number of frame called frequency and equal to 15 in the code implemented during this thesis. (The choice of 15 frames will be explained later)

Tracking an item consists in following it during several frames [51]. First the object must be detected then the tracking algorithm learns part of the object features and use it to find the item in the next frame. More information is learnt about the object while the number of frame increases and this additional information is used to follow the item during several frames. During the tracking process, new objects that enter the field of view are not detected.

Because an object can disappear from the field of view or new objects can arrive at the intersection, a new detection stage must be performed. A tracking process follows this detection stage and the process described above is repeated until the end of the video. The figure below describe the succession of detection and tracking stages that are done until the end of the video.

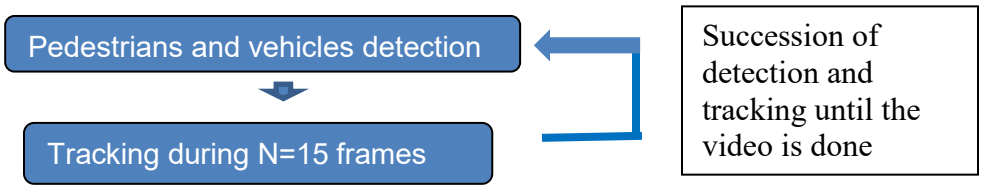


Figure 22: Two main steps of the tracking algorithm

Once pedestrians and vehicles are tracked, their location within successive frames is extracted and stored into tables. A table store on the y-axis the frame number, on the x-axis are the ID of each pedestrians or vehicles detected. The value contained is the table are tuple: the first value of the tuple is the x-coordinate of the detected item into pixel coordinate and the second value is its y-coordinate into pixel coordinates. The origin of the axes is at the top left corner and x-axis goes down and y-axis goes right. The picture below describe how the axes are set up.

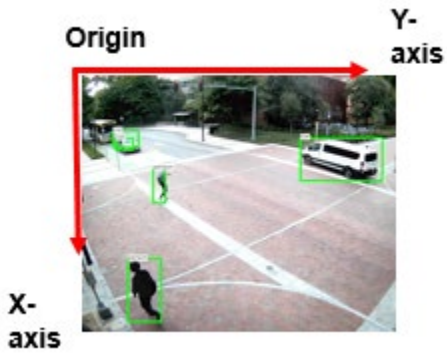


Figure 23: Origin and x and y axis

These tables, such as the one represented below are then used in a first time to determine the vehicle speed and a potential speeding.

Table 5: Sample of the location table for pedestrians at the intersection between Techwood drive and Fifth street NW at lunch time.

Frame/person ID	2	3	4	5	6
123	(418, 119)	(398, 215)	(504, 417)	(441, 118)	(466, 113)
124	(416.5, 119.0)	(405.5, 210.0)	(507.5, 417.0)	(439.0, 117.0)	(465.0, 113.0)
125	(414.5, 119.0)	(410.5, 207.0)	(500.0, 417.0)	(437.5, 118.0)	(465.5, 113.0)
126	(413.0, 119.0)	(418.0, 202.5)	(504.5, 413.5)	(436.5, 117.5)	(464.5, 113.0)
127	(412.0, 119.5)	(425.0, 198.5)	(500.5, 416.5)	(434.5, 118.0)	(463.0, 113.0)
128	(409.5, 120.0)	(432.0, 197.5)	(505.5, 420.0)	(434.0, 117.5)	(463.5, 113.5)
129	(407.0, 119.5)	(436.0, 193.5)	(503.0, 416.0)	(433.0, 118.5)	(461.5, 112.5)

In addition to these steps, the traffic light color and walking signal color are detected using computer vision for color detection. This information is then compared to the pedestrian location and to the vehicle location to determine whether there is am instance of jaywalking or red light running. The figure below is an example of pedestrians jaywalking at the intersection between Ferst drive and Atlantic drive around 4pm. The output of the algorithm implemented to detect jaywalking is a table that stores the ID of the person that jaywalked and the corresponding time of the video. The time is in seconds from the beginning of the video.

Figure 24: pedestrians jaywalking on campus

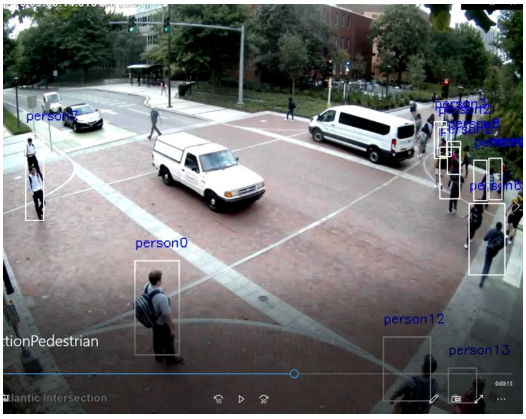


Table 6: Output of the jaywalking detection algorithm

Time sec	PersonID
1.44	2
1.44	5
2.01	5
2.01	6
2.01	9
2.025	5

Summary of steps

- The methods used to perform the persons and vehicles detection and tracking and the method used to detect the traffic and walking signal colors are computer vision techniques that will be described in details below.
- The information retrieved by the computer vision algorithms implemented during this thesis is stored into tables that form a database.
- Information from these databases are then fused to identify violations. The data is structured, which makes its analysis easier.

Hence computer vision techniques, big data (database management) and data fusion are used to build a tool that detects violations automatically and stores it into a database.

In a first subpart the vehicle and pedestrians detection will be described, then the tracking will be described. In a third subpart the light color detection will be explained.

Then the data fusion that leads to violation identification will be treated and finally the results and the benefit of the tool will be described.

The figure below describe the steps that enable to detect hazardous situations on intersections and build the incidents database.

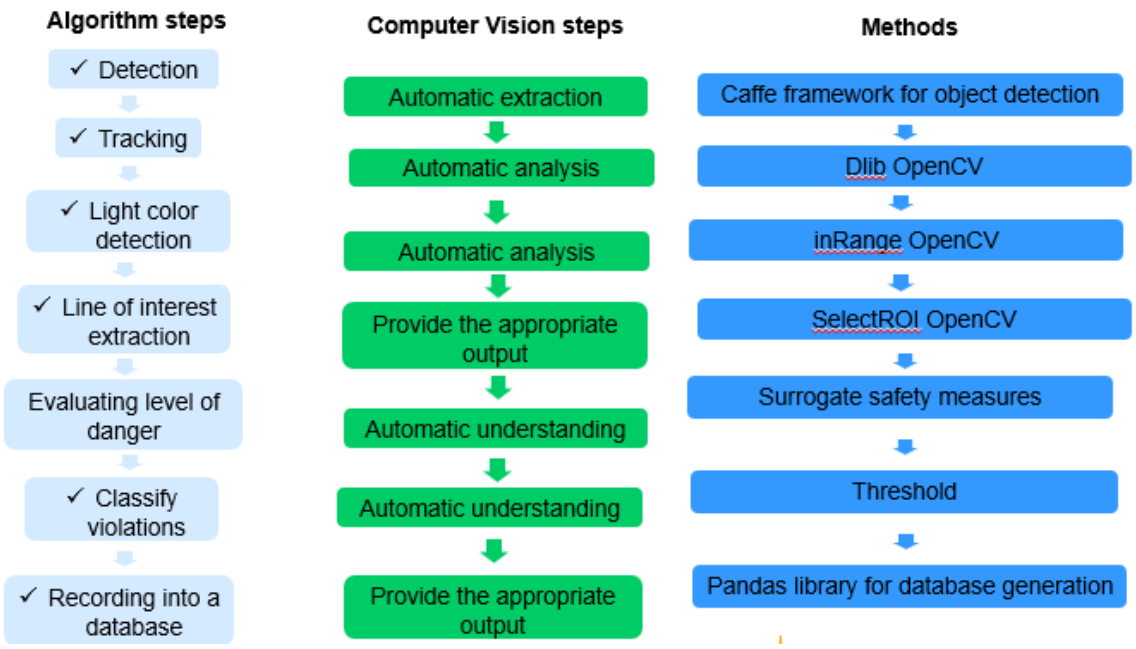


Figure 25: Steps that enable to detect and classify hazardous situations and store them into a database

5.2 Vehicle and pedestrians detection through the use of computer vision using surveillance videos

In order to perform the person and vehicle tracking on the GTPD surveillance videos, a person or a vehicle detection is first performed in the first frame.

Several objects detection methods exist such as a cascade classifier for vehicles detection, or a haar-like features algorithm for person detection and were implemented in

this thesis and compared to a convolutional neural network method. The convolutional neural network was selected as it lead to the fastest object detection once implemented on GTPD surveillance videos. The neural network used in this thesis is provided by the Caffe framework.

Bounding boxes are obtained for detected objects and the coordinates of the opposite corners of the bounding boxes are used as inputs for the tracking part.

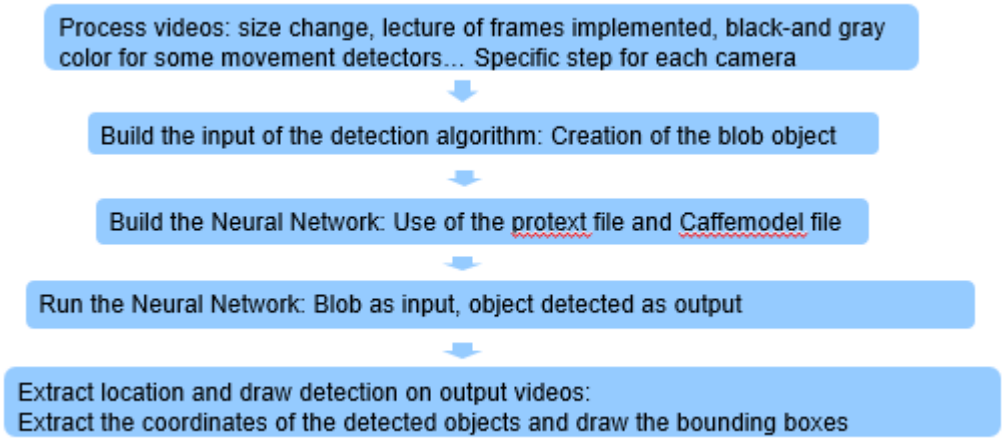


Figure 26: Steps of the detection algorithm implemented

The video is first processed so that the detection is accurate and fast. Reducing the size of the video leads to a faster processing but a less accurate detection. On the contrary, increasing the size of the video leads to the opposite observation. Hence, a compromise is achieved for the video resizing to reach an accurate and fast detection and then tracking.

Depending on the items detected a change of the pixel intensity is performed. When the detection focus on pedestrians and vehicles, using a black-and-gray color space for the video can improve the processing time and help to maintain a good accuracy.

However, if the detection focuse on the traffic light color then the video pixel intensities should be in percentage of green, red and blue (RGB images).

Once the video processing has been done, the images are converted into blob object: in the Caffe framework a blob object is the name of the standard array used as input of detection algorithms. [52]

Then, the detection convolutional neural network is built using two pretrained files from the Caffe framework: the protext file, that contains the layers architecture and the Caffemodel file, that contains the weights of the convolutional neural network.

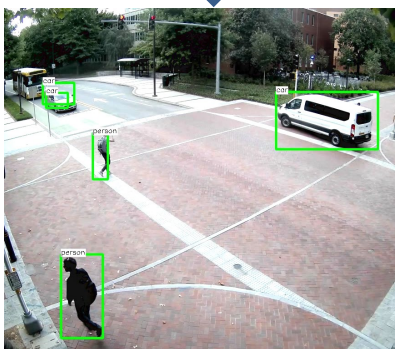
Once the neural network is built, it is run using the blob object as an input. The output is the detection of pedestrians or vehicles and bounding boxes associated with the detected item.

Finally, the coordinates of the detected items are extracted from the output and bounding boxes are drawn on the video to check the accuracy of the detection.

Pedestrians, vehicles, bicycles and buses detection using Python, Caffe and OpenCV

Surveillance video cameras provide video from campus intersections

Figure 27: surveillance video of an intersection on campus



The Caffe framework for object detection, openCV for real time computer vision and Python as the overall programming framework are leveraged to build a tool that automatically detects pedestrians, vehicles, bicycles or buses.

Figure 28: Detection performed and bounding boxes drawn

frame	class_object	CG location
1	car	[892.5, 549.5]
1	car	[808.0, 268.0]
1	car	[795.0, 368.0]
1	car	[379.5, 314.5]
1	car	[207.0, 222.0]
1	person	[276.5, 670.5]
1	person	[405.0, 591.0]
1	person	[276.5, 653.0]
2	car	[911.0, 555.0]
2	car	[804.5, 268.0]
2	car	[762.5, 368.0]
2	car	[384.5, 317.0]
2	car	[205.0, 224.0]
2	person	[276.5, 676.0]
2	person	[406.5, 588.0]
2	person	[280.0, 657.0]
3	car	[924.0, 569.0]

Once pedestrians are detected their location is computed and recorded into a database using Pandas library.

The centroid of these bounding boxes is computed and stored in a list called “coordinates” that will become a table storing location information later in the code.

Figure 29: Items detected and their location are stored into a table

The detection algorithm was first implemented for the intersection between Ferst drive and Atlantic drive. Then the detection was performed for different intersections around campus (Figure 30), such as Fifth street NW and Spring street NW, Fifth street NW and Techwood drive. Because the angle of view and the distance camera-road are different for these intersections, some settings must be changed into the code. For instance, the way to resize the video must be adapted for each intersection so that the detection of vehicles and pedestrians is accurate and fast.

Once the video is resized, the main part of the detection algorithm can be used for each intersection. Hence, the tool built can be adapted easily to other videos.

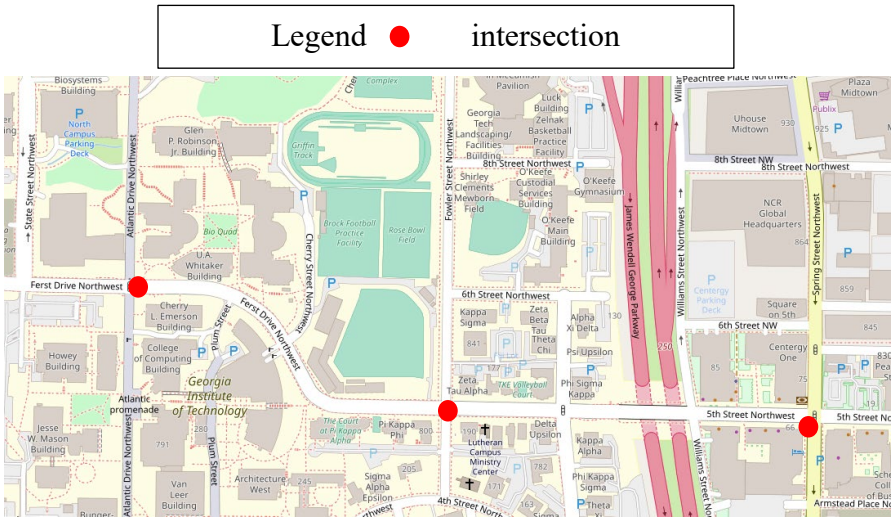


Figure 30: Georgia Tech campus and the three locations of the surveillance video cameras used in this thesis [53]

5.3 Vehicle and pedestrians tracking

Once the objects are detected in the first frame, the tracking stage of these objects starts. Two separated codes were implemented to track vehicles and pedestrians. This enables each code to run faster and in parallel. Hence, allowing for the location data to be extracted faster than if the code where run sequentially.

The dlib correlation tracker was used to implement the tracking code. The dlib correlation tracker was chosen because it is an efficient tracker to follow objects that change in both translation and scaling. [54] Because the scale of pedestrian changes significantly along their path it was necessary to use a tracker that account for scales.

Some trackers are performant for translation but lose track of objects when there is a change of scale. The dlib correlation tracker uses a scale pyramid to estimate the scale and determine in parallel the translation done by the tracked objects. [55]

In order to install the dlib library of opencv a virtual environment must be installed on Windows. Installing this environment is necessary to be able to use the tool built during this thesis. Details about the environment set up can be found in the Appendix.

Once the virtual environment is set up, the tracking algorithm can be implemented.

Below is the list of steps implemented into the Tracking algorithm:

Initialization

- Initialization of all dictionaries, data frames, lists used in the code
- Initialization of the classes to detect: persons or vehicles

- Build the Caffe detection network by using the protext and model files from the Caffe framework

- Process the video: read it frame by frame and modify the size to get an accurate and fast detection and tracking. Modify the rgb type of images into rgb ordering for the dlib library use

- Build the blob object: The blob object is the standard array in the Caffe framework and is the input of the detection network

For the first frame

- Detection and selection of objects (vehicles or persons) whose confidence is higher than 40 percent. When an object is detected using the Caffe framework, the probability of error of the class (person, bicycle, vehicle) detected is provided and called confidence.

- Store the centroid location into a coordinates dictionary which contains the objectIDs as keys and the centroid locations as values. Later in the code this dictionary is used to build a table of locations. An example of location table is shown below (Table 7).

Table 7: Sample of the vehicle locations at 10am at the intersection between Ferst drive and Atlantic drive

Frame/carsID	5	6	7	8
245	(173.5, 144.0)	(276.5, 145.5)	(83.5, 109.5)	(71.0, 107.5)

246	(175.5, 145.0)	(270.0, 143.5)	(85.5, 109.5)	(71.0, 107.5)
247	(178.5, 146.0)	(269.5, 142.5)	(85.5, 109.5)	(72.0, 108.0)
248	(181.5, 148.0)	(262.5, 140.5)	(87.0, 110.0)	(72.0, 108.0)
249	(186.5, 149.0)	(255.5, 139.5)	(86.0, 110.0)	(72.5, 108.0)
250	(188.5, 150.5)	(248.5, 137.0)	(88.0, 110.5)	(73.5, 108.0)
251	(193.0, 152.0)	(241.5, 134.5)	(88.5, 110.5)	(73.5, 108.0)
252	(195.5, 154.0)	(238.5, 133.5)	(90.0, 111.0)	(74.0, 108.5)

On this table, the number of frame represents the number of frame since the video started and can be related to the time since the video started. Multiplying the frame number by the number of frame per second which is 0.065 seconds gives the time in second since the video started. Then the real time can be computed by adding the previous time and the real time when the video started.

- Update the last indices of the detected objects so that all different objects get a different index

- Start the tracking of these objects using the coordinates of the opposite corners of the bounding boxes: “t.start_track(rgb, rect)”. These coordinates enable to identify the object that the tracker should follow. Then the tracker learns the characteristic of this object and find him within successive frames.

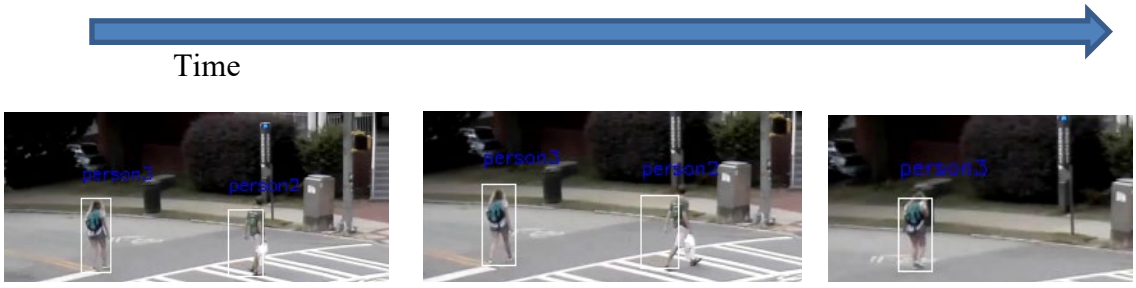


Figure 31: Tracking process enables to follow items within successive frames

- Store the tracked object into a dictionary which contains the objectID as keys and the tracking reference as values. This dictionary will be used in the steps described below.

- Print the detection to check the accuracy of the algorithm.

For the next n = 15 frames

- Update the tracking of each tracked object whose reference was stored into the tracking dictionary

- Print the tracking to check the accuracy of the algorithm

- Add the new coordinates of the objects to the coordinates dictionary

For each frame that is a multiple of n=15:

- Perform tracking as before and store the updated coordinates

- Build 'OldCoord', a dictionary with the coordinates of all tracked objects in this frame and then build a np.array with the coordinates that is needed for the comparison of coordinates dicussed later

- Perform a new detection to detect objects that left the field of view and the ones that arrived in the field of view and start the tracking of these *new detected objects*

Figure 33: Frame right before the new detection

Figure 32: Frame right after the new detection

- Store the location of the centroid of these new detected objects into a np.array (this format is necessary for the comparison explained below)

- Use the two np.array to compute the distances between each pair of old objects (objects coming from the tracking and from the detection that was performed N=15 frames before) and new objects (objects coming from the detection in the current frames). The images of figure 33 and 32 gives example of *old object* and *new objects*. Indeed the results of tracking and detection are different even if they are close. So there

is a need to match both results to identify objects that stayed in the field of view, objects that left the field of view and objects that arrived in the field of view.

- Matching process: The Euclidean distance is used to match a new detected object with an old tracked one.

The minimum distances between old and new objects are computed

- If for the new detected object there are no closer old objects then both objects are matched.
- If there are no associated objects then the old object is deleted or the new object added to the list.

The chart below associated with the table provide an example of this matching process. The green points are the new objects and the orange ones are the old objects.

Table 8: Chart and output of the matching process used to implement the tracking algorithm

Table 9: Matching result between tracked object and detected ones

Object ID	Associated with	Left	Entered
1		yes	no
2	5	no	no
3	6	no	no
4		no	yes
5	2	no	no
6	3	no	no

During the steps the coordinates dictionary with the objectsID as keys and the coordinates as values is updated.

The objects that are not detected anymore and that have disappeared for at least p=3 detection stages are removed. The value of p can be modified, several trials were done and increasing p leads to a poorer detection because some objects are detected late.

The new objects are added to the dictionary with a new objectID.

- Start the tracking of the new detected objects and the ones still here
- Print the tracking to check the accuracy of the algorithm

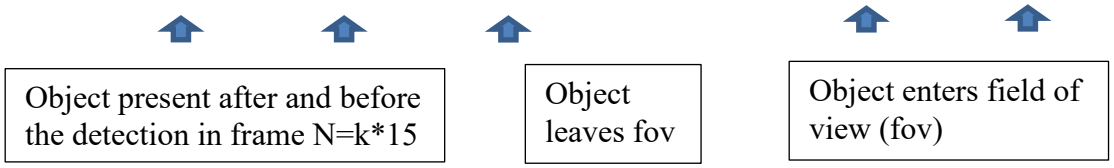
Tables created

- Build Dataframes that contains the location information stored so far into dictionaries.

The output of the tracking process is a table that contains the location of each vehicle (respectively each pedestrian) for each frame. The location is a tuple with the first value being equal to the x-coordinate and the second value the y-coordinates into pixels. Each 15 frames a new detection is performed where new object are detected, object that left the field of view are identified and other objects are tracked. The Table 10 shows the output of this tracking and matching process.

Table 10

Frame/Person	Person 0	Person 1	Person 2	Person 3	Person 4	Person 5
14	(93.0, 228.0)	(55.0, 245.5)	(393.5, 169.0)	(278.0, 164.0)		
15	(90, 230)	(52, 244)		(281, 165)	(14, 248)	(101, 234)
16	(91.5, 230.0)	(54.0, 245.0)		(282.5, 164.5)	(16.5, 248.0)	(102.0, 234.0)



The chart below summarize the two main steps of these algorithms: detection and tracking and represents the final output: coordinates tables for pedestrians and vehicles.

Step 1. detection

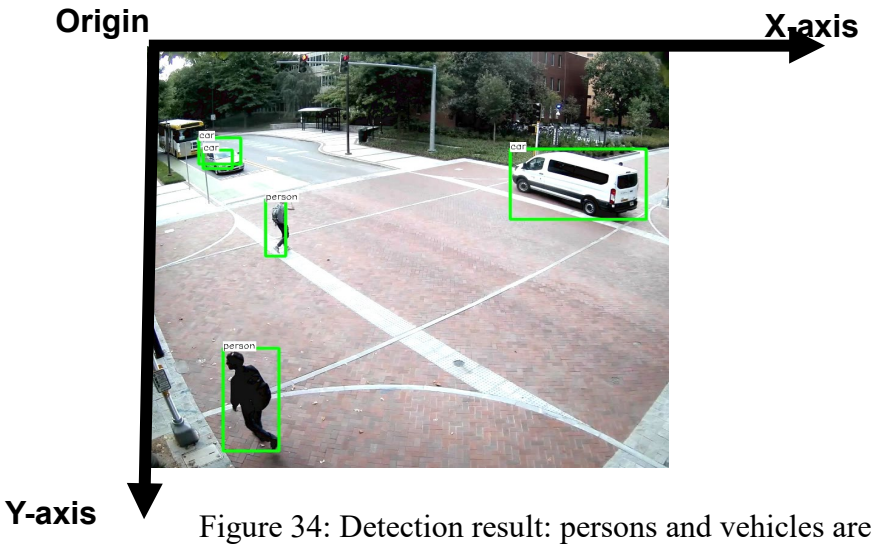


Figure 34: Detection result: persons and vehicles are detected

Step 2. tracking

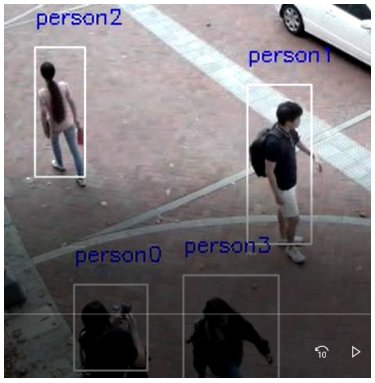


Figure 35: Pedestrian tracking output

Pedestrian ID

Location in pixel coordinates

time	0	1	2	3	4	5
0	(140.5, 412.5)	(217.0, 314.5)	(155.5, 325.5)	(154.5, 372.0)		
0.09	(140.5, 412.5)	(219.0, 314.5)	(152.5, 324.0)	(157.5, 376.0)		
0.18	(141.0, 412.5)	(221.0, 314.5)	(149.0, 321.0)	(160.5, 377.5)		
0.27	(141.0, 412.0)	(222.0, 315.0)	(147.0, 321.0)	(164.5, 378.5)		
0.36	(142.0, 412.5)	(224.0, 315.0)	(142.5, 316.0)	(168.5, 381.0)		
0.45	(142.5, 412.5)	(226.0, 315.0)	(139.0, 314.0)	(172.5, 383.5)		
0.54	(143.0, 413.0)	(228.5, 315.0)	(135.5, 309.5)	(176.5, 386.0)		
0.63	(143.5, 412.5)	(230.0, 315.5)	(132.5, 306.5)	(181.0, 390.0)		
0.72	(144.5, 412.5)	(232.5, 315.0)	(131.0, 304.5)	(186.0, 394.5)		
0.81	(146.0, 413.0)	(235.0, 315.5)	(127.5, 303.5)	(190.5, 395.5)		

Figure 36: Pedestrian Location Table

Figure 38: Vehicle tracking output

Vehicle ID

⇩

time	0	1	2	3	4	5
0	(473.5, 268.0)	(204.5, 168.5)	(436.0, 145.5)	(431.0, 193.0)		
0.09	(489.5, 273.0)	(210.0, 170.0)	(436.0, 145.5)	(420.0, 190.0)		
0.18	(509.0, 282.0)	(211.0, 172.5)	(436.0, 145.5)	(415.0, 186.0)		
0.27	(517.5, 290.0)	(219.5, 174.5)	(436.0, 145.0)	(402.0, 183.0)		
0.36	(537.5, 298.5)	(223.0, 177.0)	(436.0, 145.0)	(394.0, 180.0)		
0.45	(555.0, 306.5)	(228.0, 180.0)	(436.0, 145.0)	(384.0, 178.0)		
0.54	(567.0, 314.5)	(233.0, 182.5)	(436.0, 145.0)	(382.5, 175.0)		
0.63	(586.5, 324.5)	(241.5, 185.0)	(436.0, 145.0)	(369.5, 172.5)		

Figure 37: Vehicle location table

Choice of N=15 frames between two detections:

The number of 15 frames between two detections was chosen because 1 frame represents 0.065 seconds. So 15 frames represents 0.975 so about 1 second. If the average speed for a pedestrian is 1.4m/s and the average speed for cars is 20km/h so 5.6m/s then the pedestrians have moved from less than 2 meters since the last detection and vehicles from less than 5.5 meters.

The field of view diagonal is about 15 meters so each new person or vehicle that enter the field of view within this 15 frames will still be in the field of view during the following detection step.

Regarding the pedestrians, no information is lost by assuming N=15 because the violations made will still be detected after 15 frames since the maximum length walked by them is 2 meters. However, some information might be lost about the vehicle tracking.

Reducing the number of frames between two detections decreases the risk of error but increases the processing time as detection is heavier to compute than tracking. The number N=15 frames was a good compromise between accuracy and processing time.

The algorithm steps implemented for the tracking of pedestrians and vehicles and the recording of their locations into tables, is summarized in the chart Figure 39.

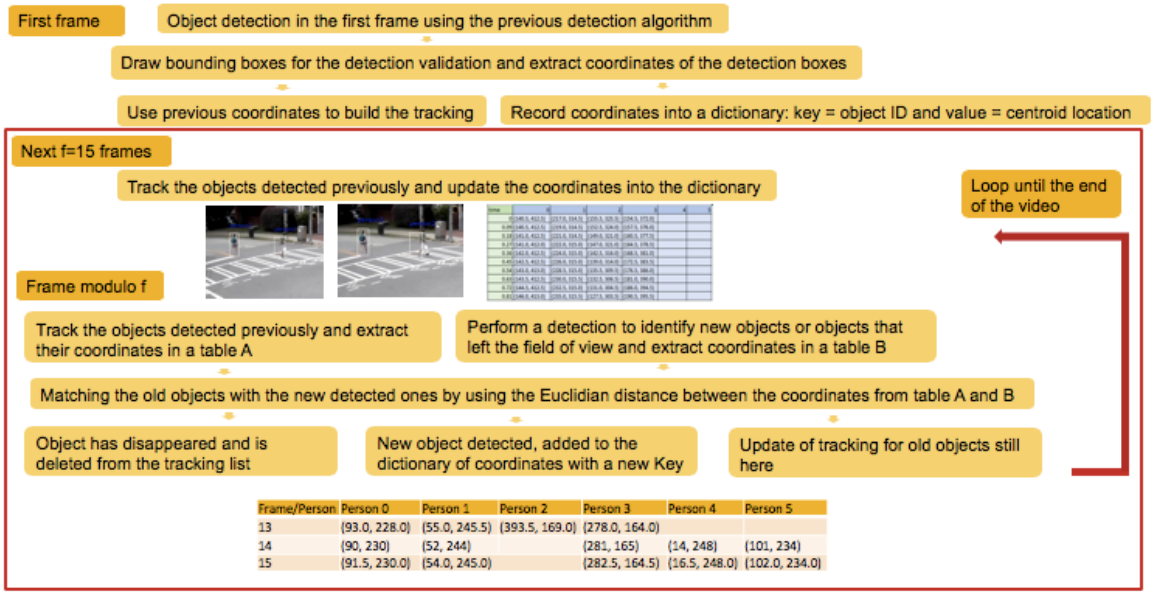


Figure 39: Summary of the detection and tracking algorithms

The output of this algorithm is a location table for vehicles (respectively pedestrians) that contains the coordinates of each vehicle (respectively each pedestrians) passing around the intersection. The Figure 40 shows an example of output.

Frame	Cars 6	Cars 7	Cars 8	Cars 9
29	(387.0, 123.0)	(454.0, 115.5)	(471.5, 119.0)	(163.0, 368.0)
30	(390, 122)		(470, 119)	(94, 414)

31	(387.5, 122.0)		(470.5, 119.0)	(101.0, 412.0)
32	(388.0, 122.0)		(470.0, 119.0)	(99.0, 409.0)

Figure 40: Example of vehicles tracking output for Fifth street NW and Techwood drive NW intersection

5.4 Speeding detection

Once trajectories are recorded into tables, the vehicle speed can be determined and compared to a threshold to identify speeding. The speeding detected is then recorded into a table that contains the time when the violation occurred and the ID of the vehicles that went above the speed limit.

- First the location table is used to compute the speed. (Figure 42)

  $Vx = \frac{X(t+dt)-X(t)}{dt}$ and $Vy = \frac{Y(t+dt)-Y(t)}{dt}$ where dt=0.065sec is the time between two frames. The picture below remind the directions of axis and how X and Y are defined.

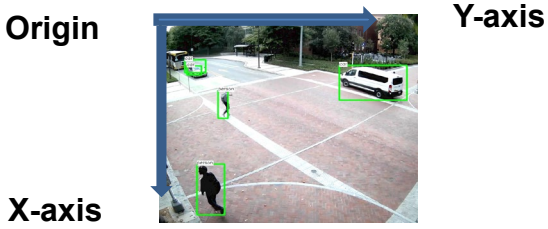


Figure 41: Origin and axes directions of the coordinate frame

- The speed table with the speed in pixel/seconds is then obtained. A positive speed corresponds to a vehicle going to the right direction and a negative speed to a vehicle going to the left. (Figure 43)
- The speed is then converted into km/h and the direction and the norm is stored into a table. (Figure 44)

To convert the speed from pixel/seconds to km/h the following approximation is made: The length of the diagonal of the image corresponds to 15 meters.

Because the images were resized to a number of pixel along the x axis equal to 600 and a number of pixel along the y axis equal to 449, the diagonal contains 749 pixels and represents 15 meters. Hence 1 pixel represents 0.02 meters. Using the relationship between pixel and meter, the speed in km/h is computed and stored into a table.

To determine whether a speeding occurs the speed is compared to a speed limit equal to 50km/h which corresponds to the speed limit around Campus.

The three Figures 42, 43 and 44 describe the steps detailed previously and show the format of the input (location tables, speed tables etc) and outputs used in the algorithm.

frame	time	0	1	2	3
0	0	(473.5, 268.0)	(204.5, 168.5)	(436.0, 145.5)	(431.0, 193.0)
1	0.065	(489.5, 273.0)	(210.0, 170.0)	(436.0, 145.5)	(420.0, 190.0)
2	0.13	(509.0, 282.0)	(211.0, 172.5)	(436.0, 145.5)	(415.0, 186.0)
3	0.195	(517.5, 290.0)	(219.5, 174.5)	(436.0, 145.0)	(402.0, 183.0)
4	0.26	(537.5, 298.5)	(223.0, 177.0)	(436.0, 145.0)	(394.0, 180.0)

Figure 42: Location table

frame	time	Car 0	Car 1	Car 2	Car 3
2	0.13	(246, 76)	(84, 23)	(0, 0)	(-169, -46)
3	0.195	(300, 138)	(15, 38)	(0, 0)	(-76, -61)
4	0.26	(130, 123)	(130, 30)	(0, -7)	(-200, -46)
5	0.325	(307, 130)	(53, 38)	(0, 0)	(-123, -46)
6	0.39	(269, 123)	(76, 46)	(0, 0)	(-153, -30)
7	0.455	(184, 123)	(76, 38)	(0, 0)	(-23, -46)
8	0.52	(300, 153)	(130, 38)	(0, 0)	(-200, -38)
9	0.585	(307, 146)	(69, 53)	(0, 0)	(-100, -38)
10	0.65	(253, 123)	(138, 46)	(0, 7)	(-123, -38)
11	0.715	(38, 115)	(76, 53)	(0, 0)	(-169, -53)
12	0.78	(30, -353)	(138, 53)	(0, 0)	(-38, -30)
13	0.845	(-38, 76)	(130, 53)	(0, 0)	(-169, -38)
14	0.91	(-7, -38)	(100, 61)	(0, 0)	(-92, -46)
15	0.975	(0, 0)	(200, 15)	(-46, 38)	(-161, 38)

Figure 43: speed Vx and Vy into pixel coordinate frame

frame	time	Car 0	Car 1	Car 2	Car 3	Car 4
2	0.065	40.53801	28.27062	0	34.61069	0
3	0.13	45.7757	24.94144	0	29.01658	0
4	0.195	34.88559	31.606	0	36.77597	0
5	0.26	46.00409	26.69548	0	31.45506	0
6	0.325	43.29667	28.39626	0	33.22577	0
7	0.39	37.93544	28.11788	0	25.70293	0
8	0.455	46.2469	31.75168	0	36.65762	0
9	0.52	46.47629	28.26441	0	29.70232	0
10	0.585	42.25466	32.47346	0	31.269	0

Figure 44: Speed table in km/h

A table that contains the time of the speeding and the index of the vehicle is created (Table 45). If the algorithm is associated with a license plate recognition algorithm then citations can be sent automatically.

Figure 45: Sample of the speeding detection table for the intersection between Atlantic drive and Ferst drive around 4pm

frame	time	CarID
25	1.625	8
26	1.69	8
27	1.755	8
64	4.16	17
65	4.225	17

5.5 Traffic light and walking signal color detection

In order to automatically detect automatically jaywalking or red light running, both traffic light and walking signal colors must be determined automatically. To perform the traffic light color detection the function “inRange” of OpenCV (computer vision framework) is used: this function enables the detection of object whose pixel intensity values are within a specified range.

Lower bounds and upper bounds for the Red, Blue, Green values

In a first time, the boundaries for the Red, Green and Blue (RGB) values for the Red traffic light, Green traffic light and Orange traffic light must be determined. Indeed taking the intensities (255, 0, 0) for the red, which corresponds to the theoretical RGB value of the red color, did not lead to a consistent result in this work. It is due to the fact that the video has a background color slightly green and the colors intensity value differs from their theoretical value.

Through several trials the following boundaries were used for the red detection, orange detection and green detection (Table 11). The first value corresponds to the blue intensity, the second to the green and the third to the red because OpenCV represents images as Numpay arrays of Blue, Green, Red pixels (BGR instead of RGB).

Table 11: Lower and upper bounds for color detection on the GTPD surveillance videos

Color	Orange	Red	Green
Lower bound	[10, 100, 20]	[25, 255, 255]	[33, 80, 40]
Upper bound	[17, 15, 100]	[50, 56, 200]	[102, 255, 255]

Traffic lights image cropping

Once the lower bound and upper bounds for Red, Green and Orange colors in the GTPD videos have been determined, the traffic lights are cropped from the initial image. The new images hence obtained contain traffic light color and a small part of the surrounding background only. This helps to avoid the detection of vehicle colors or other items color that are not the traffic lights into the global image.

Because the detection code detects every red or orange color items, it is important to crop the image to focus on the traffic light color only.

Red and Orange detection

The openCV function inRange was used to detect red and orange colors. It takes as input the image where we want to detect the color (the traffic lights cropped from the original frame), and two boundaries for the R, G, B value of the detected color.

Once the detection has been performed, the result is stored into a table (Table 12) containing the frame number, the corresponding time since the video started and three columns for each color green, red and orange, that contains 1 if their light is on and 0 if it is off.

Table 12: Sample of the traffic light color table

Time (sec)	frame	Green	Orange	Red
0.065	1	1	0	0
0.13	2	1	0	0
0.195	3	1	0	0
0.26	4	1	0	0
0.325	5	1	0	0
0.39	6	1	0	0
0.455	7	1	0	0
0.52	8	1	0	0
0.585	9	1	0	0
0.65	10	0	1	0
0.715	11	0	1	0
0.78	12	0	1	0

While the algorithm is running and the color detection is performed, the detected light color is written on the video. This enables to check whether the output is true compared to the real color seen on the video or if there is any error. (Figure 47)

Figure 46: Light color output checked by writing the output into the video and comparing it to the real color

Once the right lower bounds and upper bounds for red and orange were determined, the detection ended up being accurate.

While the detection of both red and orange color was accurate, the detection of the green light led to inaccurate results, with the traffic light being detected as green when it actually was not.

Problem with the green color detection

The inaccuracies in the green color detection are assumed to be caused by the image background that is slightly green. In order to improve the green detection the following steps were implemented:

- Cropping more accurate: The traffic light cropping was done such that only the green light was. Doing so helped reduce errors due to leaves in the background. As some trees are located close to the light. However, the detection was still not 100% percent accurate after improving the cropping. The light was said to be green when it was actually off and another light (red) was on. It is probably due to the fact that the background color of the image is slightly green.

- Conversion of the image from RGB into HSV: HSV images are known to have less noise when it comes to color information. Several filters used to detect the green color were combined to improve the detection. It enabled to improve the detection but it was still inaccurate sometimes.

As a work-around it was assumed that if the light was not red or orange then it was green. However, this assumption leads to a problem in the case where the traffic light is broken. Again the output of the green light detection is written on the video to check its accuracy. (Figure 48)

A potential solution would be to focus on the brightness detection of the traffic light. Hence, if the light located at the lower level is bright, it can be concluded that the light is green.

Figure 47: Traffic light color detection output

The same process was performed to detect the walking signal color.

It could be thought that the walking signal color is complementary from the traffic light color but it is not true as there is a short time during which both lights are red. It enables pedestrian to finish crossing the street before vehicles start. A sample of a walking signal color detection is presented in the table 49.

time	frame	Green	Orange	Red
0	1	0	0	1
0.065	2	0	0	1
0.13	3	0	0	1
0.195	4	0	0	1
0.26	5	0	0	1
0.325	6	0	0	1
0.39	7	0	0	1

Figure 48: Pedestrian walking signal color detected

5.6 Pedestrian and vehicle count

Once pedestrian and vehicle locations are determined, the flows at the intersection can be computed.

Three different counts were performed in this thesis. The first one provides the number of vehicles and pedestrians present at the intersection without considering if they are exiting or arriving. It can be used by police officers to determined needs for an officer to manage traffic or crossing. Indeed if, for a given intersection, the flows are very high every morning at 9am then the Police department can deploy agents at the intersection to help managing flows.

A second count implemented provides the arriving and exiting flows at the intersection. Finally a third one, provides the arriving and exiting flows coming or leaving each street that intersects. These counts can be used as input to a simulation environment such as SUMO to calibrate the simulation with real data.

First count: number of vehicles and pedestrians at the intersection

This count is performed by querying the location table to get the number of items present. The query checks whether a location is provided or whether the corresponding box is empty and count the number of filled columns.

Tables 13 and 14 are obtained, the first one for the pedestrian count and the second one for the vehicle count.

Table 13:Vehicle count table

frame	time since the video started (s)	count
15	0.975	4

45	2.925	4
75	4.875	2
105	6.825	3
135	8.775	4
165	10.725	4
195	12.675	3
225	14.625	2
255	16.575	2
285	18.525	2
315	20.475	2
345	22.425	3
375	24.375	1
405	26.325	1
435	28.275	1
465	30.225	1

Table 14: Pedestrians count table

frame	time since the video started (s)	count
15	0.975	4
45	2.925	3
75	4.875	5
105	6.825	2
135	8.775	2
165	10.725	2
195	12.675	2
225	14.625	2
255	16.575	2
285	18.525	3
315	20.475	2
345	22.425	1
375	24.375	1
405	26.325	0

435	28.275	0

The output of these tables are plotted to visualize the evolution of pedestrians and vehicles flow evolution at the intersection.

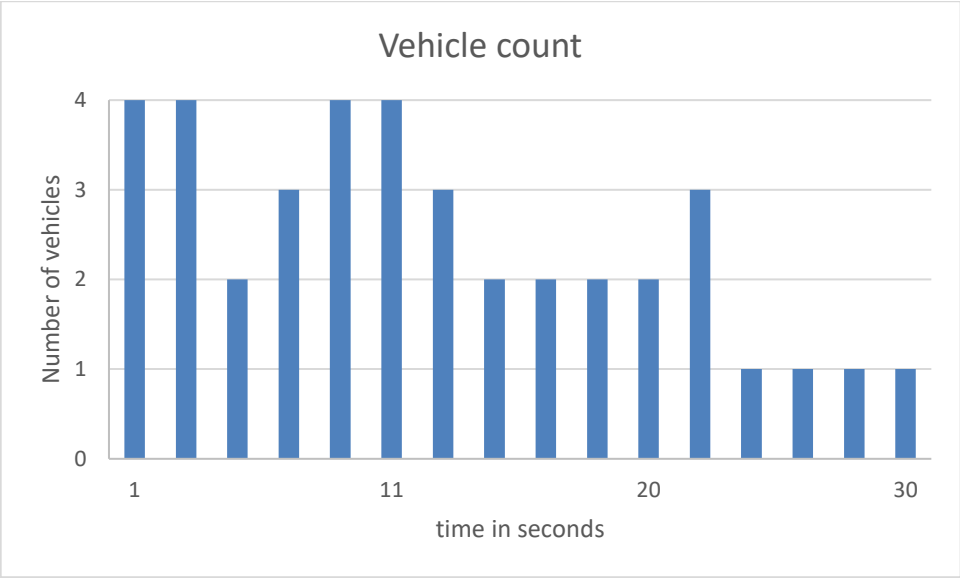


Figure 49: Vehicle count as a function of time (in second)

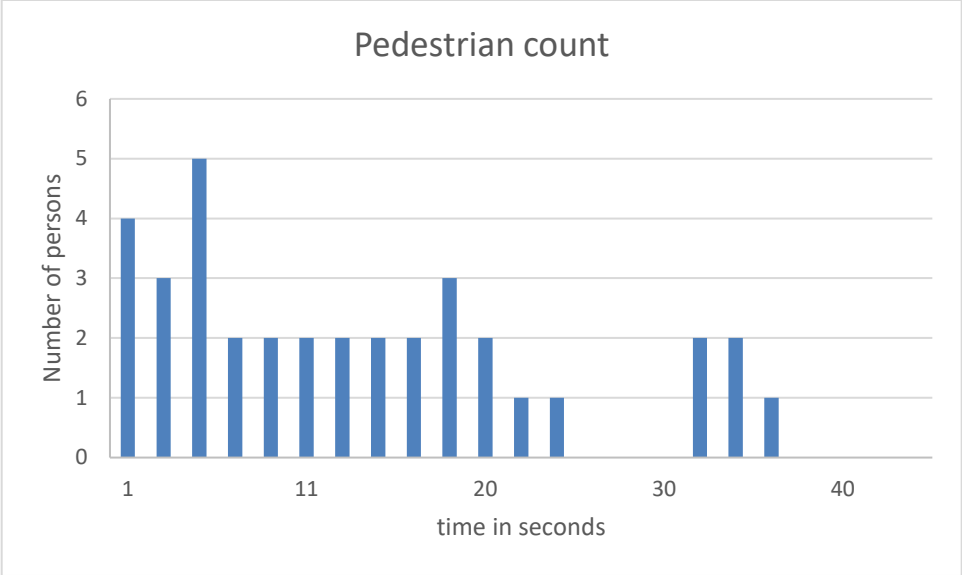


Figure 50: Pedestrian count as a function of time (second)

Second count, arriving and exiting flows at intersection:

In order to perform this count, the query used is a little more complex. The input are the location tables and the output is a table that provides for each frame the number of pedestrians arriving at the intersection and the number of pedestrians exiting the intersection. Tables 15 and 16 are example of outputs. The same work is done for the vehicle arriving and exiting the intersection.

Table 15: Arriving and exiting pedestrian flow at intersection between Atlantic drive and Ferst drive during lunch time

frame	Time (s)	exiting	arriving
15	0.975	0	0
30	1.95	1	0
45	2.925	0	0
60	3.9	0	2
75	4.875	0	0

90	5.85	0	2
105	6.825	1	0
120	7.8	1	0
135	8.775	0	2
150	9.75	3	0

Table 16: Entering and exiting vehicle flows at the intersection between Techwood and fifth street NW

frame	Time (s)	exiting	arriving
15	0.975	0	0
30	1.95	0	1
45	2.925	1	0
60	3.9	0	0
75	4.875	0	3
90	5.85	3	0
105	6.825	0	0
120	7.8	0	0

The algorithm was run for the three intersections: Atlantic drive and Ferst drive, Techwood and Fifth street NW and Fifth and Spring street NW. This result was used to calibrate the simulation of the Grand Challeng MoTIFE which aims is to build a platform that predicts traffic and incidents around campus. [67]

Details about the query is given below in a pseudo-code.

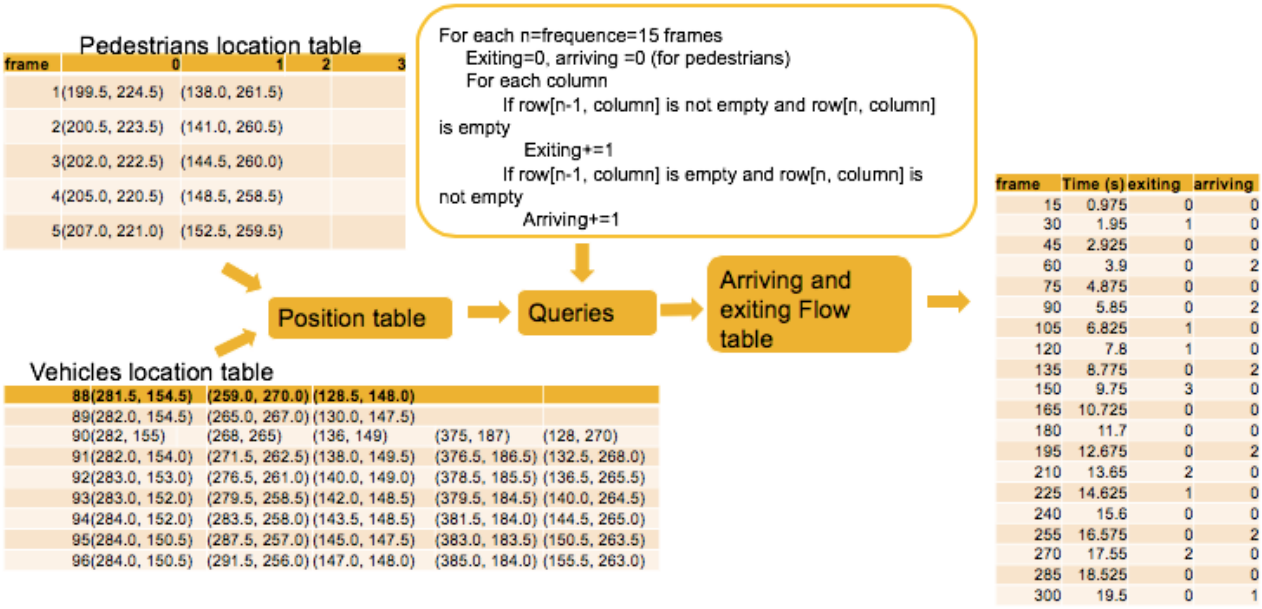


Figure 51: Arriving and Exiting flows process

Third count: arriving and exiting flows for each street that intersects:

This flow requires more time to implement as it necessitates to determine the equation of stop lines that depends on the camera location and so on the intersection. Consequently the code developed is intersection-specific and needs to be adapted if to be used at a different intersection.

This is further described in the second part of this thesis, it the one that focus on traffic light optimization, as it was implemented to calibrate the simulation model as part of the traffic light optimization effort.

5.7 Detection of the lines of interest

Once the locations of pedestrians and vehicles are obtained in the pixel coordinate frame, it is necessary to determine the equation of lines of interest such as the crosswalks demarcations or the stop lines for vehicles.

These equations are used to determine whether the pedestrian is located on the crosswalk or steps into traffic or to determine whether the vehicle crossed the stop line or not. The figure 52 describes this process.

This information is then compared to the traffic light signal to determine whether the pedestrian or the vehicle is in a violation or not.

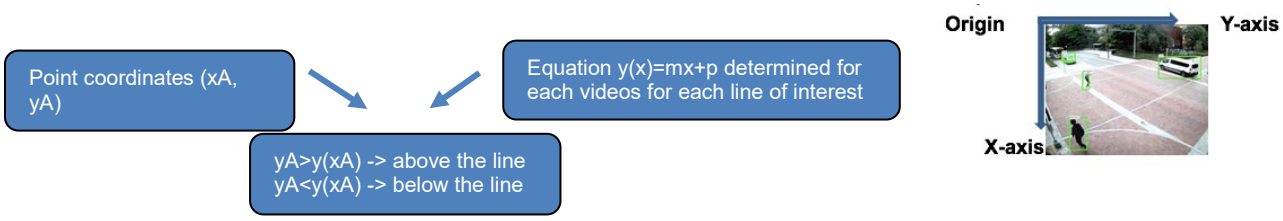


Figure 52: Process used to determine whether the pedestrians is on the crosswalk or not

To determine these equations the coordinates of two points of each line must be extracted from the video. To do so a code was implemented, that enables to select any area of interest on the video while the video is running and extract the coordinates of the selected area or point. The figure 53 shows how this code is used on a video.

Figure 53: Screen shot of the video running with the algorithm than enables to select any area of interest (blue box) and extracts its location

Once two points per line of interest are extracted from the video, the line equation is determined and stored into a table so that it is easy to use as input of another code.

Table 17: Example of output for the two previous codes for the intersection of Atlantic drive and Ferst drive.

Coordinates of chosen points extracted	Equation of line computed
corner1B = (129, 165) corner2B = (269, 135)	yBack= int((corner2B[1]-corner1B[1])/(corner2B[0]-corner1B[0])*(x-corner1B[0])+corner1B[1])
corner3F = (582, 250) corner4F = (514, 393)	yFront= int((corner3F[1]-corner4F[1])/(corner3F[0]-corner4F[0])*(x-corner4F[0])+corner4F[1])

In table 17 yBack is the equation of the stop line for vehicles coming from the upper part of the image and yFront is the equation of the stop line for vehicles coming from the bottom part of the video. Corner 1B and 2B are the two green points at the back on the figure 54 and corner 3F and 4F are the two yellow points at the front on this figure.

In order to check whether the lines equation are correct, these lines are drawn on the video as shown below. The left picture shows the points extracted with the algorithm and the right pictures shows the crosswalk demarcation lines.

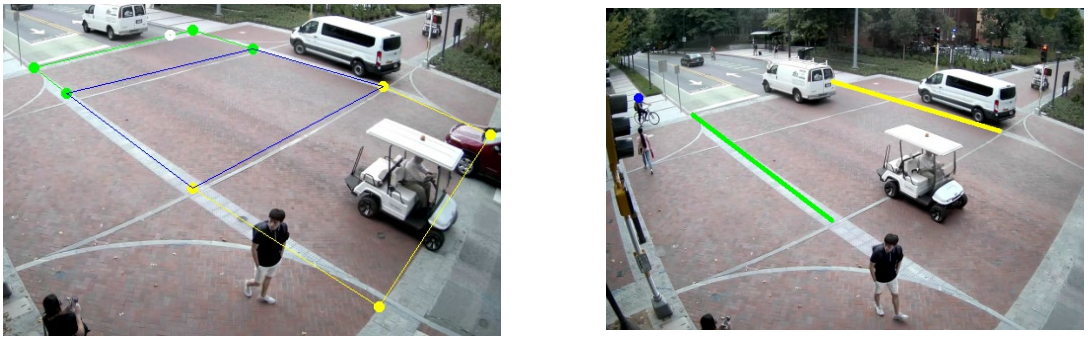


Figure 54: Stop lines and crosswalk demarcation are drawn on the video to check the rightness of their equation

5.8 Data fusion for violation detection

As discussed, this research focuses on the following violations: jaywalking, red light running, speeding, near collisions.

In order to detect these violations the data extracted previously needs to be merged. This is represented in Figure 55. The inputs gathered so far are represented in green and the output that is the violations detected are in blue.

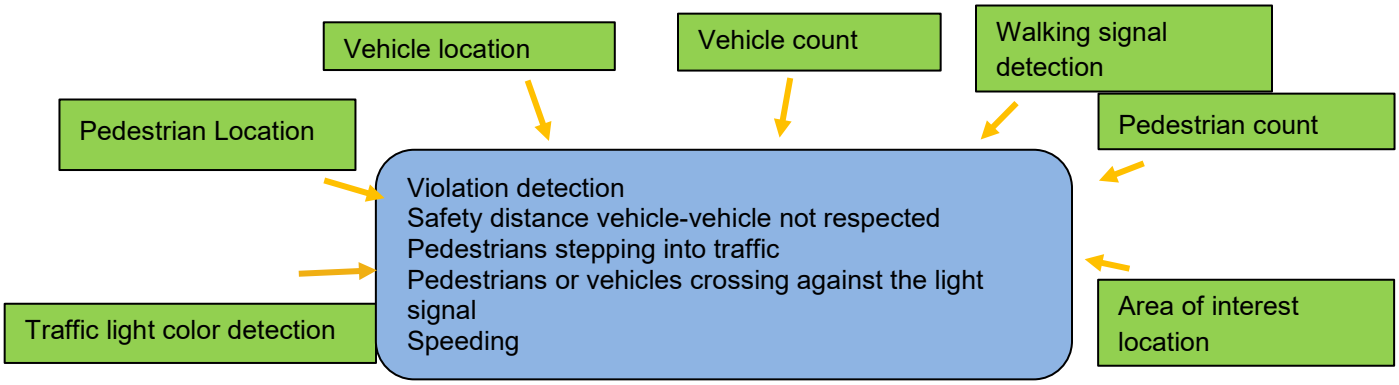


Figure 55: Data fusion process

Jaywalking:

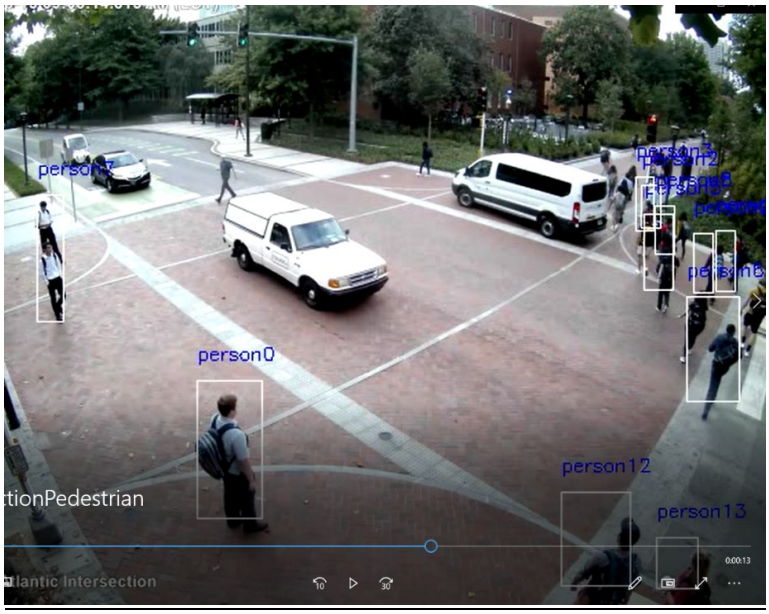


Figure 56: jaywalking example

In order to determine whether a person is jaywalking, data about the walking signal color, the pedestrian location and the crosswalk demarcation are merged.

In a first step the persons are detected and tracked and their location is recorded into a table that stores the location of each person in a given frame, i.e. at a given time.

The frame number is used rather than the time itself because the frame number is an integer while the time is a float and it is more accurate to compare two integers together than comparing two floats.

Table 18: pedestrian location table

Person_ID/ Frame	0	1	2	3	4
1	(122.0, 345.0)	(380.0, 198.0)	(542.5, 282.5)	(462.5, 181.0)	(495.0, 233.5)
2	(123.5, 346.5)	(381.5, 197.0)	(543.0, 280.0)	(464.0, 180.5)	(489.0, 231.5)
3	(124.5, 349.0)	(384.0, 196.0)	(543.5, 277.5)	(465.0, 179.5)	(481.5, 230.0)
4	(126.5, 351.0)	(387.0, 195.5)	(543.5, 275.5)	(466.5, 179.0)	(474.0, 229.0)

In a second step the walking signal light is detected and stored into a table that contains the frame number and 1 in the column that corresponds to the light color that is on and 0 in the other columns.

frame	Green	Orange	Red
1	0	0	1
2	0	0	1
3	0	0	1
4	0	0	1
5	1	0	0

Table 19: Walking signal table

For a given frame, if the walking signal is set to red, i.e. the Red column in table 19 contains 1, then each pedestrian location is analyzed automatically. The figure 57 shows these steps.

frame	Green	Orange	Red
1	0	0	1
2	**0**	**0**	**1**
3	0	0	1
4	0	0	1
5	1	0	0
6	1	0	0

Person_ID/ Frame	0	1
1	(122.0, 345.0)	(380.0, 198.0)
2	**(123.5, 346.5)**	**(381.5, 197.0)**

Figure 57: Steps followed for the jaywalking or red light running detection

Each pedestrian location is compared to the crosswalk demarcation equation (Figure 58) to determine whether the pedestrian steps into traffic.

```
(P[1] < int((corner1B[1]-corner4F[1])/(corner1B[0]-corner4F[0])*(P[0]-
corner4F[0])+corner4F[1])) and (P[1] > int((corner2B[1]-corner3F[1])/(corner2B[0]-
corner3F[0])*(P[0]-corner3F[0])+corner3F[1]))
```

Figure 58: Sample of code that determine whether a pedestrian steps into traffic

If the walking signal is green for the given frame and some pedestrians are jaywalking then the index of these pedestrians is recorded into a table that also provides the time of jaywalking (Table 20).

Table 20: Output of the jaywalking detection algorithm

Time sec	PersonID
1.44	2
1.44	5
2.01	5
2.01	6
2.01	9
2.025	5

As demonstrated, the tool built as part of this thesis enables the detection of jaywalking events and their record into a table. If the code is run on a large number of hours and during different semesters, statistics about jaywalking per semester can be obtained by querying the output table. From there the need for countermeasures to avoid jaywalking can be determined.

Red Light Running:

In order to determine whether a vehicle runs a red light (Figure 59) or not, data about the traffic light color, the vehicle location and the stop line demarcation are merged.

The steps are similar to the ones described during the jaywalking detection: first the vehicle location is extracted from the video and stored into a table, then the light color is detected and stored into a table.

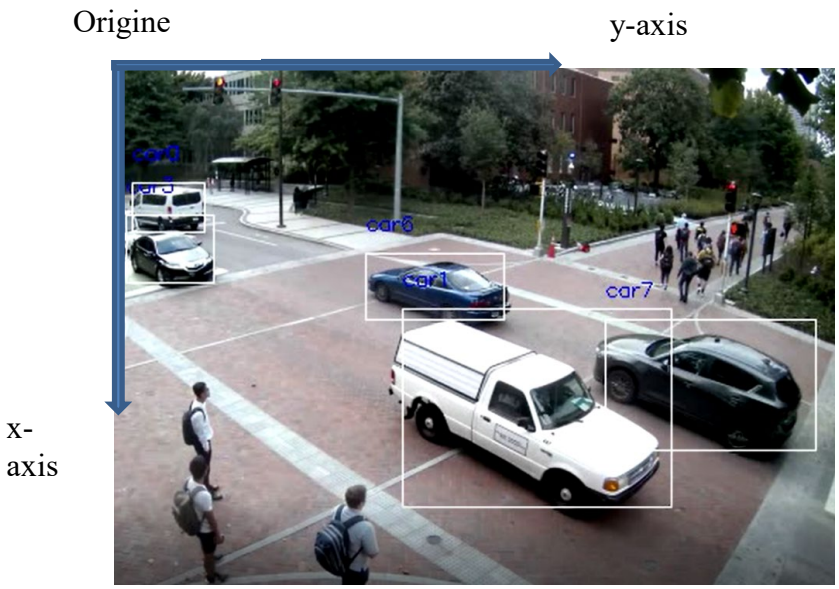


Figure 59: vehicles running a red light (GTPD surveillance video at intersection of Atlantic drive and Ferst drive NW)

If the light is red then the location of each vehicle is analyzed and compared to the stop line equation. The tables 21 and 22 describe this steps.

Table 21: Sample of the traffic light table

frame	Green	Orange	Red
120	**0**	**0**	**1**
121	0	0	1
122	0	0	1
124	0	0	1
125	0	0	1

Table 22: Sample of the vehicle location table

Cars_ID/ Frame	0	1
120	**(122.5, 200.0)**	**(79.0, 165.5)**
121	(120.5, 201.0)	(81.0, 165.5)
122	(118.5, 202.0)	(79.0, 165.5)
123	(129, 205)	(82, 166)

```
P[1] < int((corner3F[1]-corner4F[1])/(corner3F[0]-
corner4F[0])*(P[0]-corner4F[0])+corner4F[1])) and (P[1] >
int((corner2B[1]-corner1B[1])/(corner2B[0]-corner1B[0])*(P[0]-
corner1B[0])+corner1B[1])
```

Figure 60: Sample of the code that shows the stop line equations

If the vehicle crosses the stop line while the light is red (Figure 60 shows the lines equations used to determine such violation) then its index is stored into a table along with the time of the violation. It enables to record all red light running automatically and it can be used to make enforcement operations automatic. The table 23 is a sample of output provided by the algorithm that detect red light running.

Table 23: Output table for the red light running detection

Time sec	Vehicle_ID
28.26	1
28.26	6
28.26	7
28.35	1
28.35	6
28.35	7

If this tool is associated with an analytic tool able to detect plate number then citations can be sent automatically as soon as a vehicle is running a red light. It could help the Police department to enforce the red light running around campus.

Safety distance vehicle-vehicle not respected:

Collisions are dangerous situations that can be detected through the use of data extracted by computer vision. In this work, vehicles that don not respect the safety distance between vehicles are detected and their index is sorted into a table.

First, the direction of each vehicle is determined using the speed table in pixel coordinates (Figure 61) . In this table, negative values correspond to a vehicle going from right to left and positive values to a vehicle going from left to right.

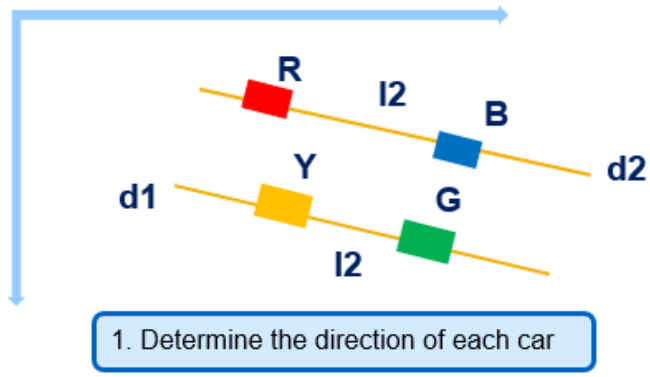


Figure 61: Determine the direction of each car

In figure 61 the yellow lines represent the two directions at the intersection between Ferst drive and Atlantic drive. The rectangles are vehicles.

Once the direction is determined, the distance between two vehicles going in the same direction is determined by first determining the vector that connect these two vehicles (Figure 63) and then computing its length (Figure 62).

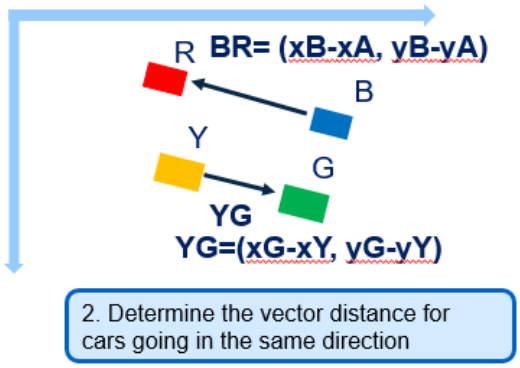


Figure 63: Determine vectors using coordinates of vehicles

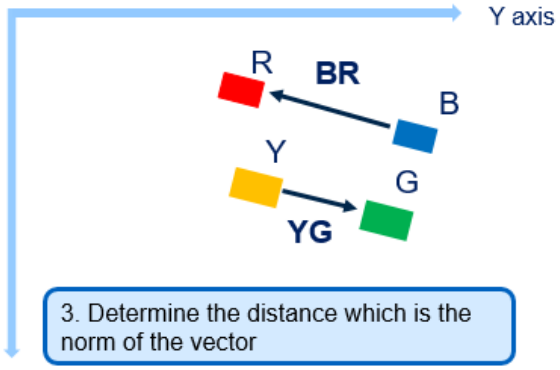


Figure 62: Determine the distance between vehicles

Then the distance is compared to a safety limit that was taken to 5 meters or 16 feets. The safety distance recommended between two vehicles corresponds to the distance that is crossed by the vehicle during 3 seconds. If we assume an average speed of 40km/h then the safety distance is 33 meters or 108 feets. However this distance was often not respected at this intersection according to the results of the computer vision techniques applied on the videos. So a safety distance of 5 meters (177 inches) which is an average for the vehicle length was chosen. (Figure 64)

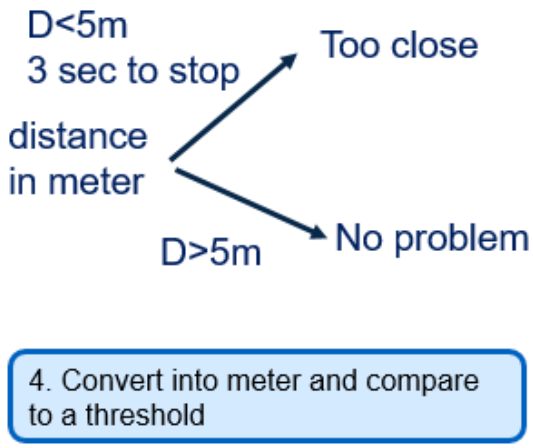


Figure 64: Vehicle-vehicle distance compared to a safety distance equals to 5m

The result is then stored in a table (Table 24) that contains the time of the 'violation' and the ID of the vehicles that drove too close to each other's.

Table 24: Output table that stores the time and the indices of the vehicles that drove to close to each other

frame	Time (seconds)	Index such that <Dlimit
14	0.91	(3, 4)
29	1.88	(3, 4)
37	2.40	(3, 4)
39	2.53	(3, 4)
40	2.6	(3, 4)
42	2.73	(3, 4)
105	6.82	(2, 3)
119	7.73	(6, 5)
120	7.8	(6, 5)
121	7.86	(6, 5)
122	7.93	(6, 5)

Note: It is necessary to determine the direction of the vehicles because the image is a 2D image so the depth is not known. Hence, the left corners of vehicles going from left to right can coincide with the left corner of vehicles going from right to left. The algorithm would then detect a collision whereas there is no collision because the two vehicles are in different lines.

The Figure 65 depicts this problem when the left rear corner of Vehicle 1 coincides with the left rear corner of Vehicle 6 even though no collision occurred because the vehicles directions are different.

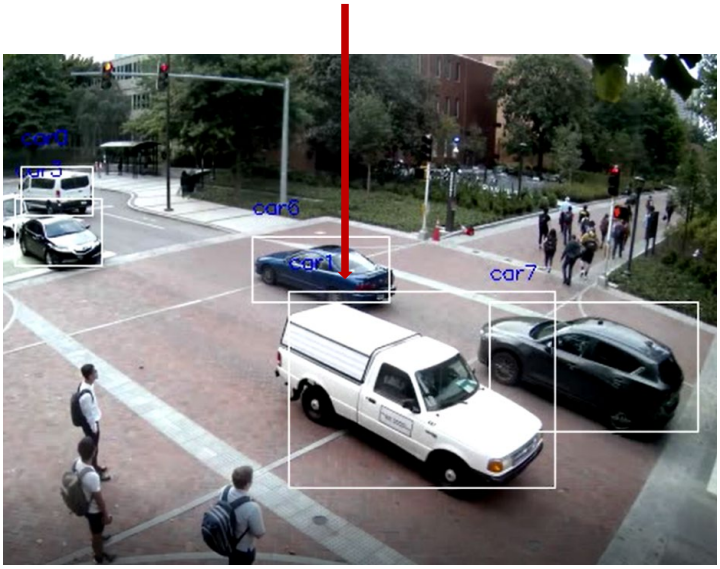


Figure 65: Two corners of different vehicles have the same coordinates

An assumption was made for this task: no vehicle is going in the wrong direction into its lane. In order to be able to perform a collision detection without accounting for the direction a second camera that could provide information about the depth.

5.9 Database and statistics (results and use by the Police)

Through the use of computer vision on the surveillance videos, data were extracted and a database was built. It gathers information for three different intersections around the Georgia Tech campus:

- Ferst drive and Atlantic drive
- Fifth street NW and Spring street NW
- Fifth street NW and Techwood drive NW

The information gathered in this database is the following:

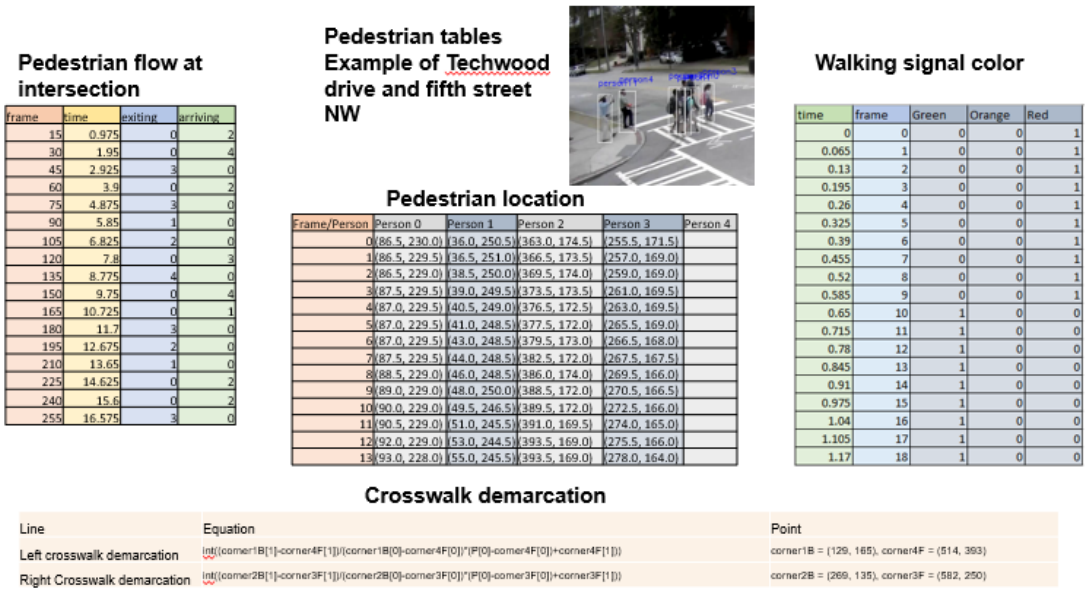
- The vehicle and pedestrian location
- The traffic light color and walking signal color
- The crosswalks demarcation and the stop lines location
- Vehicles speed
- Jaywalking instances
- Red light running instances
- Speeding instances
- Situations with a high risk of collisions because vehicles are too close
- The pedestrians and vehicles counts at intersections

The database can be divided into three parts, as illustrated in Figures 66, 67 and 68.

- The pedestrian part: pedestrian location, pedestrian flow and crosswalk demarcation

- The vehicle part: vehicle location, vehicle flows and stop line location

- The infraction detection part: speeding, jaywalking, red light running, high risk of collision due to small distance vehicle-vehicle.



Pedestrian flow at intersection

frame	time	exiting	arriving
15	0.975	0	2
30	1.95	0	4
45	2.925	3	0
60	3.9	0	2
75	4.875	3	0
90	5.85	1	0
105	6.825	2	0
120	7.8	0	3
135	8.775	4	0
150	9.75	0	4
165	10.725	0	1
180	11.7	3	0
195	12.675	2	0
210	13.65	1	0
225	14.625	0	2
240	15.6	0	2
255	16.575	3	0

Pedestrian tables Example of Techwood drive and fifth street NW

Pedestrian location

Frame/Person	Person 0	Person 1	Person 2	Person 3	Person 4
0	(86.5, 230.0)	(36.0, 250.5)	(363.0, 174.5)	(255.5, 171.5)	
1	(86.5, 229.5)	(36.5, 251.0)	(366.5, 173.5)	(257.0, 169.0)	
2	(86.5, 229.0)	(38.5, 250.0)	(369.5, 174.0)	(259.0, 169.0)	
3	(87.5, 229.5)	(39.0, 249.5)	(373.5, 173.5)	(261.0, 169.5)	
4	(87.0, 229.5)	(40.5, 249.0)	(376.5, 172.5)	(263.0, 169.5)	
5	(87.0, 229.5)	(41.0, 248.5)	(377.5, 172.0)	(265.5, 169.0)	
6	(87.0, 229.5)	(43.0, 248.5)	(379.5, 173.0)	(266.5, 168.0)	
7	(87.5, 229.5)	(44.0, 248.5)	(382.5, 172.0)	(267.5, 167.5)	
8	(88.5, 229.0)	(46.0, 248.5)	(386.0, 174.0)	(269.5, 166.0)	
9	(89.0, 229.0)	(48.0, 250.0)	(388.5, 172.0)	(270.5, 166.5)	
10	(90.0, 229.0)	(49.5, 246.5)	(389.5, 172.0)	(272.5, 166.0)	
11	(90.5, 229.0)	(51.0, 245.5)	(391.0, 169.5)	(274.0, 165.0)	
12	(92.0, 229.0)	(53.0, 244.5)	(393.5, 169.0)	(275.5, 166.0)	
13	(93.0, 228.0)	(55.0, 245.5)	(393.5, 169.0)	(278.0, 164.0)	

Walking signal color

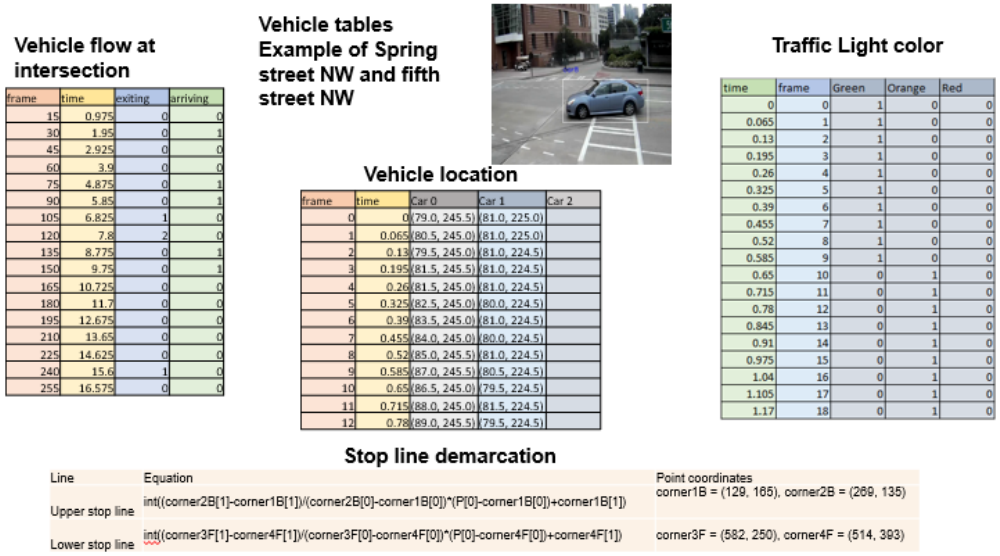
time	frame	Green	Orange	Red
0	0	0	0	1
0.065	1	0	0	1
0.13	2	0	0	1
0.195	3	0	0	1
0.26	4	0	0	1
0.325	5	0	0	1
0.39	6	0	0	1
0.455	7	0	0	1
0.52	8	0	0	1
0.585	9	0	0	1
0.65	10	1	0	0
0.715	11	1	0	0
0.78	12	1	0	0
0.845	13	1	0	0
0.91	14	1	0	0
0.975	15	1	0	0
1.04	16	1	0	0
1.105	17	1	0	0
1.17	18	1	0	0

Crosswalk demarcation

Line	Equation	Point
Left crosswalk demarcation	int((corner1B[1]-corner4F[1])/(corner1B[0]-corner4F[0])*(P[0]-corner4F[0])+corner4F[1])	corner1B = (129, 165), corner4F = (514, 393)
Right Crosswalk demarcation	int((corner2B[1]-corner3F[1])/(corner2B[0]-corner3F[0])*(P[0]-corner3F[0])+corner3F[1])	corner2B = (269, 135), corner3F = (582, 250)

Figure 66: Pedestrian database overview

Figure 67 : Vehicle database overview



Vehicle flow at intersection

frame	time	exiting	arriving
15	0.975	0	0
30	1.95	0	1
45	2.925	0	0
60	3.9	0	0
75	4.875	0	1
90	5.85	0	1
105	6.825	1	0
120	7.8	2	0
135	8.775	0	1
150	9.75	0	1
165	10.725	0	0
180	11.7	0	0
195	12.675	0	0
210	13.65	0	0
225	14.625	0	0
240	15.6	1	0
255	16.575	0	0

Vehicle tables Example of Spring street NW and fifth street NW

Vehicle location

frame	time	Car 0	Car 1	Car 2
0	0	(79.0, 245.5)	(81.0, 225.0)	
1	0.065	(80.5, 245.0)	(81.0, 225.0)	
2	0.13	(79.5, 245.0)	(81.0, 224.5)	
3	0.195	(81.5, 245.5)	(81.0, 224.5)	
4	0.26	(81.5, 245.0)	(81.0, 224.5)	
5	0.325	(82.5, 245.0)	(80.0, 224.5)	
6	0.39	(83.5, 245.0)	(81.0, 224.5)	
7	0.455	(84.0, 245.0)	(80.0, 224.5)	
8	0.52	(85.0, 245.5)	(81.0, 224.5)	
9	0.585	(87.0, 245.5)	(80.5, 224.5)	
10	0.65	(86.5, 245.0)	(79.5, 224.5)	
11	0.715	(88.0, 245.0)	(81.5, 224.5)	
12	0.78	(89.0, 245.5)	(79.5, 224.5)	

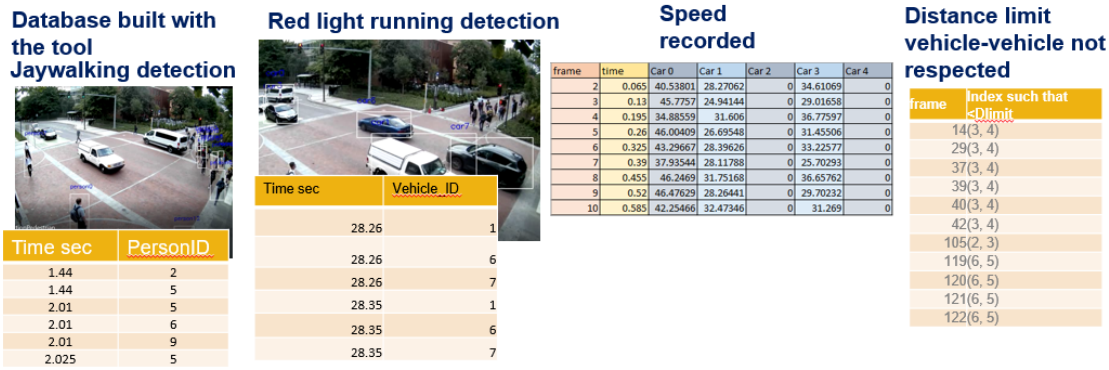
Traffic Light color

time	frame	Green	Orange	Red
0	0	1	0	0
0.065	1	1	0	0
0.13	2	1	0	0
0.195	3	1	0	0
0.26	4	1	0	0
0.325	5	1	0	0
0.39	6	1	0	0
0.455	7	1	0	0
0.52	8	1	0	0
0.585	9	1	0	0
0.65	10	0	1	0
0.715	11	0	1	0
0.78	12	0	1	0
0.845	13	0	1	0
0.91	14	0	1	0
0.975	15	0	1	0
1.04	16	0	1	0
1.105	17	0	1	0
1.17	18	0	1	0

Stop line demarcation

Line	Equation	Point coordinates
Upper stop line	int((corner2B[1]-corner1B[1])/(corner2B[0]-corner1B[0])*(P[0]-corner1B[0])+corner1B[1])	corner1B = (129, 165), corner2B = (269, 135)
Lower stop line	int((corner3F[1]-corner4F[1])/(corner3F[0]-corner4F[0])*(P[0]-corner4F[0])+corner4F[1])	corner3F = (582, 250), corner4F = (514, 393)

Figure 68: Incident database overview



Database built with the tool Jaywalking detection

Time sec	PersonID
1.44	2
1.44	5
2.01	5
2.01	6
2.01	9
2.025	5

Red light running detection

Time sec	Vehicle_ID
28.26	1
28.26	6
28.26	7
28.35	1
28.35	6
28.35	7

Speed recorded

frame	time	Car 0	Car 1	Car 2	Car 3	Car 4
2	0.065	40.53801	28.27062	0	34.61069	0
3	0.13	45.7757	24.94144	0	29.01658	0
4	0.195	34.88559	31.606	0	36.77597	0
5	0.26	46.00409	26.69548	0	31.45506	0
6	0.325	43.29667	28.39626	0	33.22577	0
7	0.39	37.93544	28.11788	0	25.70293	0
8	0.455	46.2469	31.75168	0	36.65762	0
9	0.52	46.47629	28.26441	0	29.70232	0
10	0.585	42.25466	32.47346	0	31.269	0

Distance limit vehicle-vehicle not respected

frame	Index such that <Dlimit
	14(3, 4)
	29(3, 4)
	37(3, 4)
	39(3, 4)
	40(3, 4)
	42(3, 4)
	105(2, 3)
	119(6, 5)
	120(6, 5)
	121(6, 5)
	122(6, 5)

These databases provide a better understanding of high-risk situations. Because the data is structured (sorted into tables), it is easy to collect statistics about it or plot it to identify trends.

Once the algorithms are run on a large number of hours of video, the table can be queried to obtain statistics.

Once statistics are obtained about jaywalking, speeding or red light running it is possible to identify the countermeasures required at these locations, which should help to improve the safety at intersections on campus.

This data can also be used to complete the Police record database about incidents around campus. Especially the Table about situations “close to collision” can help the Police to process faster the incident database. Indeed, currently the incidents are described by using words and sentences into the incident database (Figure 49). This way of data storage is called unstructured because it is not organized in a pre-defined manner and is not adapted to get statistics and identify patterns.

LightConditi	MannerOfCc	LocationAre	Narrative
4	6	3	Driver of vehicle 1 stated that he looked down to change songs on an iPod and the vehicle ran off the road.
4	6	3	On 07/08/2007 at 0227 hours I arrived at the intersection of Ferst Drive and Hemphill Avenue in reference to investigating a single vehicle accident. Upon my arrival I observed
4	1	3	Georgia Private Property Accident Report. I observed the driver of the white Toyota, enter the North parking deck of the North Avenue Apartments off of Centennial Olympic
4	3	1	On 08/31/2007 at approximately 00:29 hours, I was dispatched to an accident on North Avenue near Cherry Street. Upon arrival I met with the Driver #1, Kimberly DeDeaux and
4	6	4	Upon my arrival ,I observed vehicle one on top of the median at Northside Drive and North Avenue. Vehicle one appeared to have sustained severe damage to the front passe

Figure 69: Current Police record database that contains unstructured data

The collision database that can be built using this thesis sort the information in a way that makes it easy to process.

Hence the tool built during this thesis could help the Police to save time and to determine easily the needs for countermeasures to improve pedestrian safety on campus.

5.10 Answer to the first question and hypotheses validation

The first question focused on the way to extract high-risk situations automatically:

RQ1: What approach would allow for high-risk situations to be automatically detected?

The hypothesis associated with this question is the following: If computer vision techniques are applied on traffic camera data then high-risk situations can be automatically detected.

Using computer vision techniques such as person and vehicle detection and tracking or color detection is an efficient way to extract information from video. Computer vision techniques enable to get information about locations and about the environment: stop line demarcation or traffic light color.

Once these pieces of information are extracted, database management practices enable to store it in a structured way that makes the identification of patterns and the determination of statistics straighforward.

After storing the information in a convenient way, data can be fused to infer new information such as speeding, red light running, jaywalking, collisions, etc.

Hence computer vision combined with database management and data fusion enable to extract high risk situations automatically. Consequently **Hypothesis 1** is validated. Figure 70 describes the steps of computer vision used in this work.

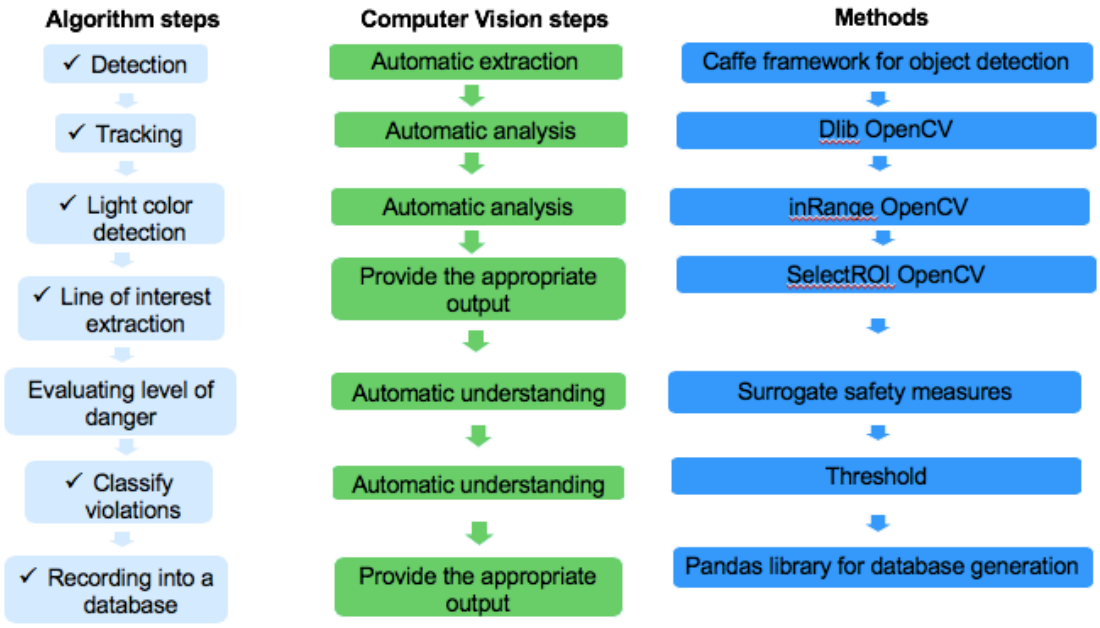


Figure 70: main computer vision techniques used in this thesis

A sub-question was raised concerning the high-risk situation identification: How are high-risk situations categorized?

Two hypotheses were determined to answer this sub-question, first, if TTC, GT and PET are used then *complex* high-risk situations can be defined and then if thresholds are applied on speed, position, pedestrian-vehicle distance, and deceleration during braking then critical traffic situations can be identified and classified.

In order to build the collision table and the speeding table distances vehicle-vehicle and speeds are compared to thresholds as described in section 5.8. In addition, in order to determine jaywalking or red light running, locations are compared to boundaries (stop line and crosswalk demarcations) that are represented by lines equations. Hence **Hypothesis**

1.2 is validated because the use of threshold on metrics such as location or speed enables the identification and classification of critical traffic situations.

Concerning Hypothesis 1.1 a limitation was encounter. Hypothesis 1.1 stated that if TTC, GT and PET are used then *complex* high-risk situations can be defined. In order to answer this hypothesis, three surrogate safety measures had to be computed. These measures give an indication of the level of danger of a conflicting situation.

Using these surrogates would allow to classify traffic situations as safe, critical or dangerous. The Post Encroachment Time (PET) provides the time difference between a road user leaving a zone and the next user arriving there, the Gap Time (GT) is the time difference between two users arriving at a confliction point and finally the Time to collision (TTC) corresponds to the time before collision if the speed and trajectory are the maintained.

The inputs necessary to compute these three parameters are the following:

- Pedestrian location
- Position of vehicle front and rear corners
- Pedestrian and vehicle velocity function
- Sidewalk demarcation
- Speed threshold
- An extrapolation function for trajectories

Most of these inputs were gathered in the work described previously, however the position of a vehicle's front and rear corners requires a third coordinate that is currently missing. For example, in a 2D coordinate system, the left rear corners of Vehicle 1 and Vehicle 6 in the Figure 71 will have the same coordinates which is not true in a three dimensional coordinate system. Hence the computation of these surrogate safety factors would be biased.

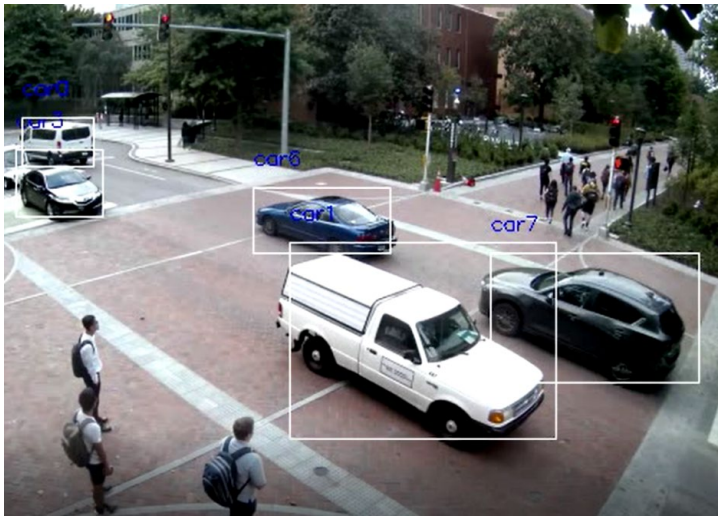


Figure 71: Vehicles 1 and 6 left rear corners have the same coordinates in a 2D frame

There is consequently a need to go from 2D to 3D. Research was done to identify a way to go from a 2D coordinate frame to a 3D coordinate one.

- The first idea was to use existing function with Matlab, e.g. the PointsToWorld function [56], or the Python function Astropy.wcs [57]. However the first function returns world points into the X-Y plane so the third coordinates can not be obtained

and the second one takes the Earth's curvature into account and works well for large distances but not for short distances.

- The second idea was to use the mathematical relation between pixels and real-world coordinates. The relation relates 3D points to their pixel coordinates into the image, using the process illustrated on Figure 72.

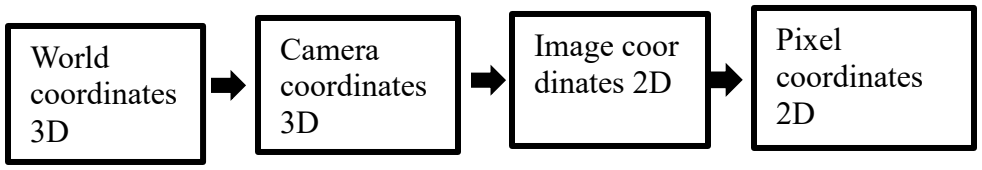


Figure 72: Steps to project 3D coordinates into the 2D pixel frame of the video

The world coordinates corresponds to a long distance coordinate frame while the camera coordinates is based on the surrounding of the camera.

The image 2D coordinates is the projection into a plane of the camera coordinates.

To go from the World 3D coordinates to the pixel 2D coordinates the following relation can be used (Figure 73). The variables into the matrices can be obtained by using camera parameters such as the focal length, the location and orientation of the cameras, etc.

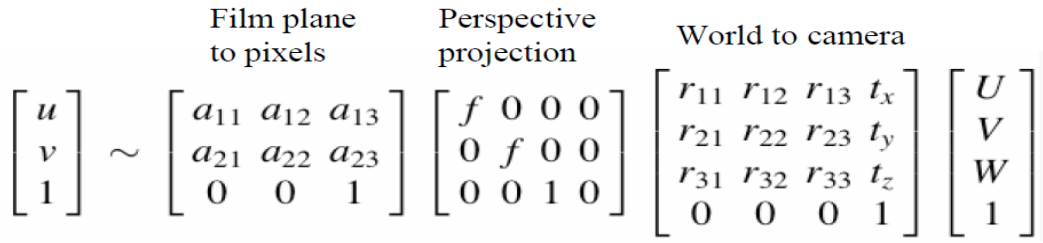


Figure 73: 3D World coordinates to pixel coordinates

This equation enables to go from the world coordinates to the pixel coordinates. Hence it needs to be inverted to get the 3D world coordinates using the pixel coordinates.

However, some information is missing to go backward, as not all parameters found in the matrices above are known by the author. Consequently, the mathematical relation can't be used to get 3D world coordinates form pixel coordinates. [58]

- The third idea was to use a second camera at the intersection in order to get more information about the location. [59] Using two cameras would enable the acquisition of the third coordinate if the cameras are properly calibrate.

Because information about the third coordinates was missing, surrogate safety indicators could not be computed. Their aim was to classify incidents as safe, critical or dangerous in order to only store in the database the incidents that are dangerous. Indeed, the storage cost increases with the storage size, so selecting the dangerous situations only would help reducing costs.

The next chapter discussed the effort done to optimize the traffic light control system at the intersection between Fifth street NW and Spring street NW. In a first part an overview

of the implementation is presented, in a second part the extraction of rush hour flows per street that intersect is presented. A third part discusses the setup of the simulation environment and finally the future work that should be done to complete the reinforcement learning optimization is presented.

CHAPTER 6. IMPROVING TRAFFIC LIGHT CHANGES AT AN INTERSECTION

When the waiting time is long at intersection pedestrians are inclined to cross the street even if the walking signal is red. Optimizing the traffic light for vehicles and pedestrians at intersection could reduce the risk of jaywalking.

The waiting time at the intersection between spring street NW and fifth street NW is significant during rush hours. This thesis focused on this intersection.

Reinforcement learning was the method selected in order to improve the traffic light signal at this intersection. This method must be run in parallel to a simulation.

In this work the simulation environment SUMO was calibrated to run the reinforcement learning algorithm. However difficulties were encountered to model correctly the pedestrian behavior and retrieve important data concerning pedestrians, consequently the reinforcement learning algorithm was not implemented entirely. Though a simple optimization algorithm that changes traffic lights following the longest vehicle queue length was implemented.

Shor summary of the Reinforcement learning process:

In a reinforcement learning algorithm an agent takes an action among a set of possible actions in order to optimize the long term reward called Q-learning function.

The algorithm attempts to learn a sequence of actions that will maximize the long term reward which is built using short term rewards from previous states. The long-term

reward is learnt when an agent interacts with an environment through many trials and errors.

A table that stores the long-term reward an agent would receive when taking this action at a particular state is created during the RL process. This table is called a Q-table.

6.1 Implementation overview

The reinforcement learning algorithm should take the pedestrians and vehicles waiting time as input and returns a green time for each phase that tends to reduce the waiting time as much as possible.

The algorithm must run in parallel to a simulation of the intersection that takes the green times as inputs and is used to determine the waiting time of vehicles and pedestrians after the green times have been schedules.

The intersection of fifth street NW and spring street NW must be implemented into the simulation environment SUMO. Then the rush hours flows at this intersection have to be extracted to calibrate the simulation.

Once it is done, the reinforcement learning algorithm must be programmed and run in parallel to the simulation.

Finally the waiting time must be retrieved from the simulation without stopping it, which requires to use a different method compared to the common one used in SUMO.

Figure 74 summarized this process.

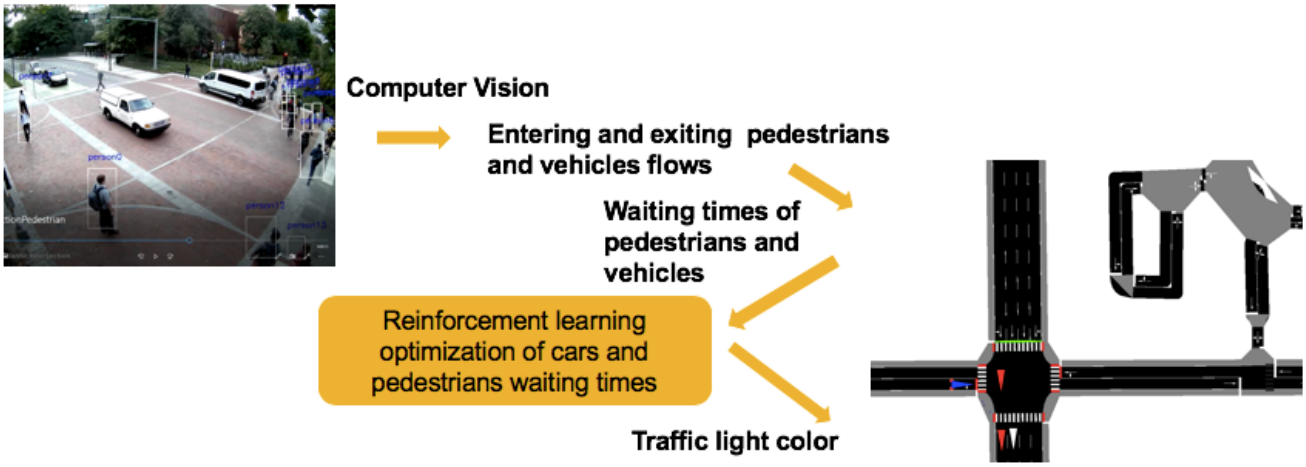
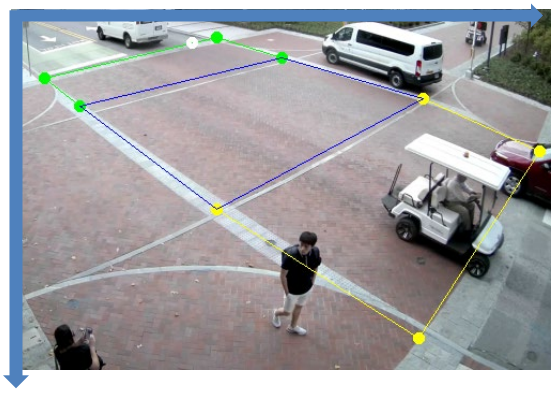


Figure 74: Simulation environment must be calibrated and then the simulation must run in parallel to the RL algorithm

6.2 Extract rush hours flows from surveillance video to calibrate the model

In order to calibrate the model then entering and exiting flows for each street during rush hours have to be implemented into SUMO. Hence these flows have to be determined. These flows were extracted from the surveillance video located at the intersection by using the method described bellow.



Let's first consider a simple street as example. Then the work done for the intersection between fifth street NW and spring street NW will be presented.

Let's consider the intersection between Atlantic drive and Ferst street where vehicles can go in only two directions.

In a first time the direction for each vehicle (Figure 76) must be determined using the speed table. Indeed the speed table (Figure 75) that contains speed into pixel coordinates takes the trip direction into account. Indeed it provides the speed with a negative sign for vehicles going up and positive for vehicles going down.

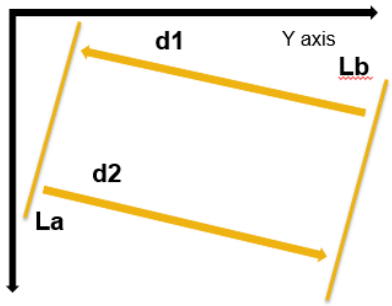


Figure 75: Chart of the intersection with two directions

frame	time	Car 0	Car 1	Car 2	Car 3
2	0.13	(246, 76)	(84, 23)	(0, 0)	(-169, -46)
3	0.195	(300, 138)	(15, 38)	(0, 0)	(-76, -61)
4	0.26	(130, 123)	(130, 30)	(0, -7)	(-200, -46)
5	0.325	(307, 130)	(53, 38)	(0, 0)	(-123, -46)
6	0.39	(269, 123)	(76, 46)	(0, 0)	(-153, -30)
7	0.455	(184, 123)	(76, 38)	(0, 0)	(-23, -46)
8	0.52	(300, 153)	(130, 38)	(0, 0)	(-200, -38)
9	0.585	(307, 146)	(69, 53)	(0, 0)	(-100, -38)
10	0.65	(253, 123)	(138, 46)	(0, 7)	(-123, -38)
11	0.715	(38, 115)	(76, 53)	(0, 0)	(-169, -53)
12	0.78	(30, -353)	(138, 53)	(0, 0)	(-38, -30)
13	0.845	(-38, 76)	(130, 53)	(0, 0)	(-169, -38)
14	0.91	(-7, -38)	(100, 61)	(0, 0)	(-92, -46)
15	0.975	(0, 0)	(200, 15)	(-46, 38)	(-161, 38)

Figure 76: table of speeds in pixel coordinates

A chart of the intersection is represented above where d1 and d2 represent the two possible directions. La and Lb represent the lines where the intersection ends (or starts). When a vehicle going into the direction d1 (respectively d2) crosses Lb (respectively La) then it arrives at the intersection. When a vehicle going into the d1 direction (respectively d2) crosses the line La (respectively Lb) then the vehicle leaves the intersection.

Once the direction of the vehicle is determined, the location of the vehicle is compared to the lines La and Lb. Hence the equation of these lines have to be determined, Table 25 presents the lines equation of La and Lb in the second column and the coordinates of the points used to find them in the first column. To get this information, the code that extracts the coordinates of points of interest and the code that compute lines equations and check their accuracy are used.

The result for the intersection is then stored into a table as tables enable to make the process automatic. Indeed once the information is stored into a table, a common code for each intersection can be used (and not separated codes).

Table 25: Lines La and Lb equation are stored into a table

Extract the location of points	**Compute the line equation using these points**
corner1B = (129, 165) corner2B = (269, 135)	La= int((corner2B[1]-corner1B[1])/(corner2B[0]-corner1B[0])*(x-corner1B[0])+corner1B[1])
corner3F = (582, 250) corner4F = (514, 393)	Lb= int((corner3F[1]-corner4F[1])/(corner3F[0]-corner4F[0])*(x-corner4F[0])+corner4F[1])

Once the vehicle direction is determined and the line equations are determined this code (pseudo code below) is used to determine the entering and exiting flow for a given street.

If d1 is true and car at the left of Lb and right of La

Then Arriving from right

If d2 true and car at the left of La

Then arriving from left

If d1 is true and car at the left of La

Then exiting at right

Or if d2 is true and car at the right of Lb

Then Exiting at left

A similar process is done for the intersection in front of Barnes and Nobles (Figure 77). However the intersection has one more possible direction to account for (Figure 78).

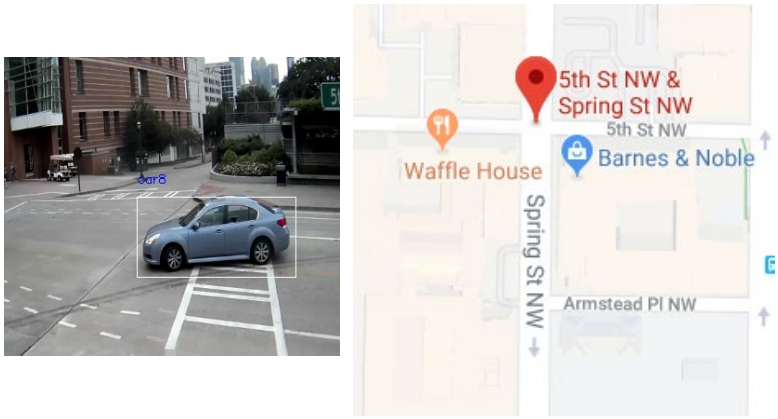


Figure 77: Barnes and Nobles intersection, view from the surveillance camera and map from open street map

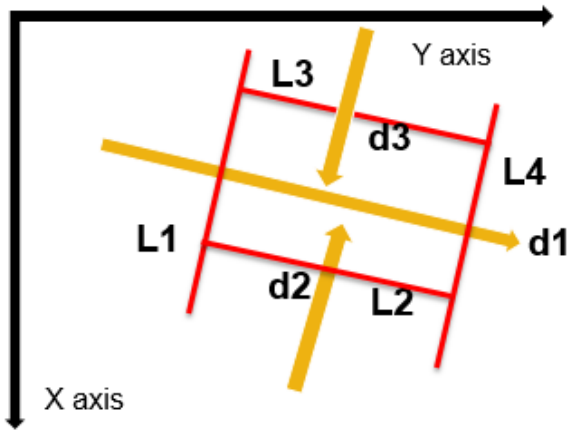


Figure 78: Three directions at the intersection in front of Barnes and Nobles

The process is similar to the previous one. However the steps to determine the vehicle directions are slightly different.

If X and Y increase then the vehicle is going in the d1 direction: In that case both components of the speed into pixel coordinates are positive.

If X increases and Y decreases then the direction is d3 and in that case the first component of the speed is positive and the second one negative.

If X decreased and Y increases then the direction is d2 and the first component of the speed is negative while the second one is positive.

The equation of lines L1, L2, L3 and L4 are defined using the two codes mentioned previously and the location of vehicles is compared to these lines equations to determine whether the vehicle is exiting or arriving at the intersection if it is crossing one of the four lines.

A sample of the result is presented in Table 26. The entering and exiting vehicles for each street during rush hours are recorder into the table at each second.

Table 26: Entering and exiting flow at the intersection for each street

frame	time	exiting street 1	arriving street 1	exiting street 2	arriving street 2	exiting street 3	arriving street 3
15	0.975	0	0	1	1	1	0
30	1.95	0	1	1	1	1	1
45	2.925	0	0	1	0	1	0
60	3.9	0	0	1	0	0	0
75	4.875	0	1	1	0	0	1
90	5.85	0	1	0	1	0	1
105	6.825	1	0	1	0	0	0

6.3 Build the simulation intersection and implement the rush hours flow

The reinforcement learning algorithm must be run in parallel to a traffic simulation in order to get the new state (new waiting times) for a given action (change of green times for traffic lights).

Hence a simulation of vehicles and pedestrians flows must be implemented. The chosen simulation environment is called SUMO and was selected because it enables to simulate pedestrian's behavior in an accurate way.

The first step in order to build the simulation was to build the map of the intersection (Figure 81). To do so the intersection of fifth street NW and spring street NW was exported from Openstreetmap (Figure 80) and converted into a set of edges and lanes so that it can be read by SUMO (Fiure 79) using a function called netConvert. [60]

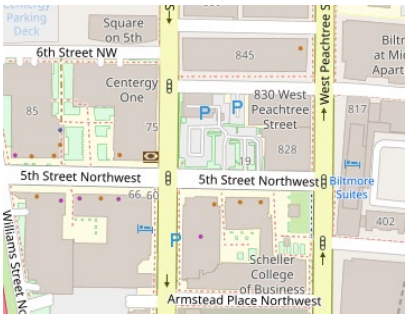


Figure 80: Intersection exported from OpenstreetMap

Figure 79: Map converted so that SUMO can read it

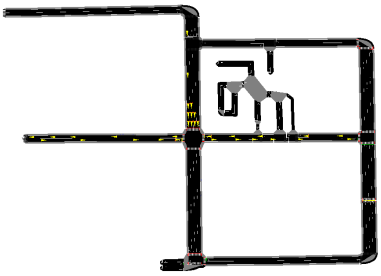


Figure 81: Intersection once SUMO reads the Map

The pedestrians and vehicles flows extracted in the previous step must be implemented to calibrate the simulation to the reality.

In order to build the vehicle flows a SUMO function called ‘flows’ can be used. However this function doesn’t exist for pedestrians. SUMO enables to create trips for pedestrians but each trip have to be created separately for each pedestrian which is time consuming. In order to overcome this difficulty a python script that enables to build a flow function for pedestrians was implemented.

The Figure 82 describes the steps of this code.

TripTime	From	To	N
(30,80)	233974204#1	234013260#4	20

For each trip in the table create N instances "persons" with random departure time within the trip duration

Sort all the persons created by increasing departure time to be readable by SUMO

Figure 82: Build pedestrian flows

6.4 Implement the optimization of the traffic light

Reinforcement learning and traffic light control problem:

For traffic light control, the environment is made by the intersection and the real vehicle and pedestrian flows. The action is a timing change of traffic lights, the state is the vehicles and pedestrians waiting times and the reward is a function of the waiting times.

Most traffic light control optimization algorithms don't take pedestrian into account. Though one paper: *Intelligent Traffic Light Control using Distributed Multi-agent Q Learning*, from Ying Liu and al included pedestrians. However in this paper the pedestrian waiting time was considered for the given intersection while the vehicle waiting time was taken into account for the given intersection and for additional intersections. Hence pedestrians might be penalize compared to vehicles.

The idea is to use a similar reward function were weights are modified to take pedestrian and vehicle into account.

The reward function described in the paper is given below.

$$R(t,i) = -w_{t,1}\frac{1}{Ni} * \sum_{j\ in\ Ni} q(j,t) + w_{t,2}\frac{1}{NiNj} * \sum_{j\ in\ Ni}\sum_{k\ in\ Nj} q(k,j,t) + w_{t,3}\frac{1}{Ni} * \sum_{j\ in\ Mi} m(j,t)$$

Where the parameters are described in the table below:

Table 27: Parameters used in the reward function

t	time
i	Intersection index
parameter	meaning
Wi,1	Weight for vehicle waiting time at intersection i
Wi,2	Weight for vehicle waiting time at all neighboring intersections
Wi,3	Weight for pedestrian waiting time at intersection i
$\sum_{j\ in\ Ni} q(j,t)$	Waiting time at intersection i
$\sum_{j\ in\ Ni}\sum_{k\ in\ Nj} q(k,j,t)$	Waiting time at each intersection around intersection i
$\sum_{j\ in\ Mi} m(j,t)$	Pedestrian waiting time at intersection i

Implementation:

During this thesis the reward function that is minimized to choose the next action was implemented however the Q-table hasn't been built yet.

To compute the reward function W1 was taken equal to ½, W2 was equal to zero because there is no information about neighboring vehicle traffic and pedestrian traffic for the intersection in front of Barnes and Nobles. And W3 is taken equal to ½.

In a future work these parameters could be changed by doing trials and it could maybe lead to a more optimized traffic control system.

Once an action is taken by an agent, the result produced by the action on the environment is evaluated and the reward is determined based on this evaluation. Hence the output of the Reinforcement learning must be connected to the simulation environment (Figure 83).

It enables to obtain the result produced by the action on the waiting time of vehicles and pedestrians and to determine the reward accordingly. The relation between the Reinforcement learning algorithm and the Simulation environment is explained in the next part.

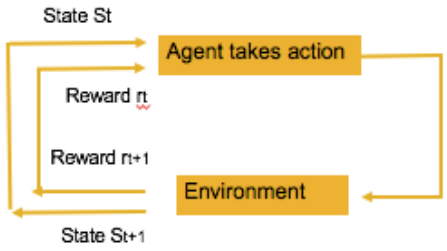


Figure 83: Interaction agent and environment modeled by SUMO

6.5 Combine Reinforcement learning and simulation

In order to implement this work, traffic light changes must be made into the simulation while it is running and waiting time of pedestrians and vehicles must be retrieved while the simulation is running.

The common way to build simulation into SUMO doesn't enable to make changes while the simulation is running. Changes have to be made before the beginning of the simulation. The traffic lights changes can be done by using NETEDIT that provides information about the SUMO network such as edges ID or traffic light sequence.

The simulation can provide the number of vehicle in each street as output. So the waiting time can be infer from this information.

However proceeding in this way require to stop the simulation to get the output or to change the traffic light sequences and green times. Hence it can't be used to implement the reinforcement learning as the simulation must keep running whiel trials are made by the agent and the traffic light should be controlled while the simulation is running.

To overcome this obstacle SUMO has developed recently and API called TracI [73] that enables to control the traffic light color while the simulation is running and to retrieve the number of vehicle stopped in a given street while the simulation runs. From this last information can be determined the waiting time of each vehicle at the intersection.

The main functions used to perform these tasks are the following functions:

- traci.edge.getLastStepHaltingNumber("EDGE ID") to get the number of stopped vehicle in a street
- traci.trafficlight.setRedYellowGreenState("cluster_2048209355_535070538_535078514_69537537", "rrrrrrrrrrrrG") where the sequence "rrrrrrrrrrrrG" is used to control the traffic light color for each light at the intersection.

Figure 84 describes which light is controlled by which part of this sequence. An index between 1 and 14 for associated to each light to make it clearer.

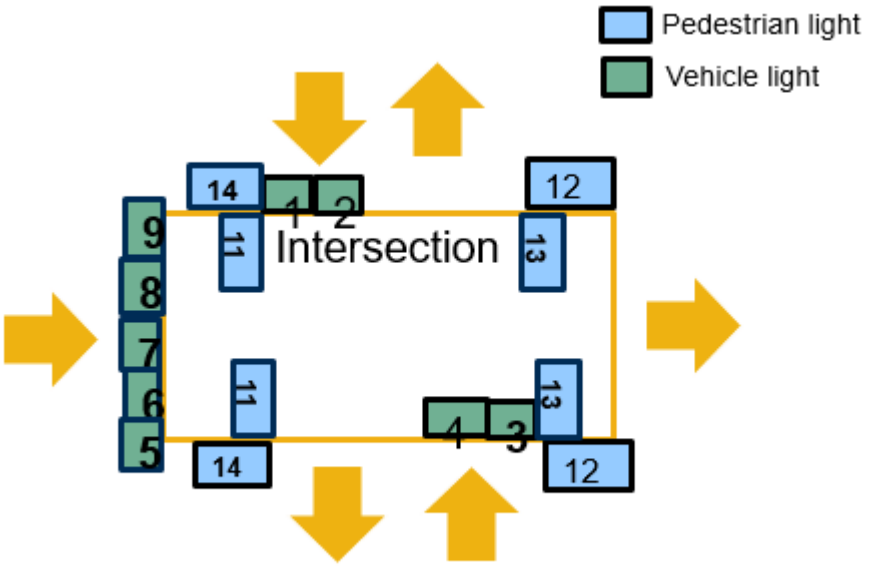


Figure 84: Light control using TracI

However TracI is a recent API and it provides a small number of function to model pedestrians. Three functions that could help retrieving the number of pedestrian waiting at the intersection are in development.

The first one is called “getWaitingTime” and is supposed to retrieve the waiting time of a given pedestrian. However the output of this function leads to a zero value for the waiting time whereas some pedestrians were waiting to cross the street.

To understand where does this error comes from the function “getSpeed” was used and it is supposed to return the speed of a given pedestrians. Then a threshold close to zero could be used to determine whether a pedestrian is stopped and so whether it is probably waiting at the crossing.

However after analyzing the output of this function it was observed that the speed retrieved was not consistent. Indeed a pedestrian stopped could have the same speed than a pedestrian moving.

To understand where does this error comes from, the position of pedestrian were obtained by using getPosition for a given pedestrian. While a pedestrian is stopped according to the simulation visualization tool and according to the traffic light situation, the position retrieved by getPosition still changed.

The Table 27 shows this observation for a pedestrian. This pedestrian is waiting at the intersection as it can be seen on the simulation visualization tool, however its location is changing.

Simulation Step	coord1	coord2
71	2673.738	2515.84
72	2672.558	2515.836
73	2671.47	2515.832
74	2670.42	2515.828
75	2669.29	2515.824
76	2668.004	2515.819
77	2666.735	2515.815
78	2665.565	2515.81
79	2664.27	2515.806
80	2662.997	2515.801
81	2661.722	2515.796
82	2660.673	2515.793

Table 28: Change in coordinate 1 while the pedestrian is stopped

Here there is an inconsistency in the function getPosition that make impossible to get the pedestrian waiting time while the simulation is running. The programmers coding for the

TracI modules warn about it by explaining that the TracI module for pedestrian is not complete yet.

A solution to this problem would be to make change to the SUMO source code, which probably re
quire several weeks.

Because of this obstacle the pedestrians couldn't be included into the model in the proper way. The following assumption were done: the average entering and exiting flows at the intersection were computing using the work described in the first part and the difference of these parameters were used as the number of waiting pedestrians.

Thus the number of pedestrians waiting is constant which biases the results.

Simulations were run using this assumption. The choice of the action were based on the longest waiting queue for vehicle and not on the Q-table. The algorithm enabled to reduce the vehicle waiting time but a better optimized traffic control system can be built with the reinforcement learning completed.

In figure 85 can be seen a long line for vehicle coming from the upper part and then a light change was decided by the algorithm that enable for these vehicles to cross the intersection.

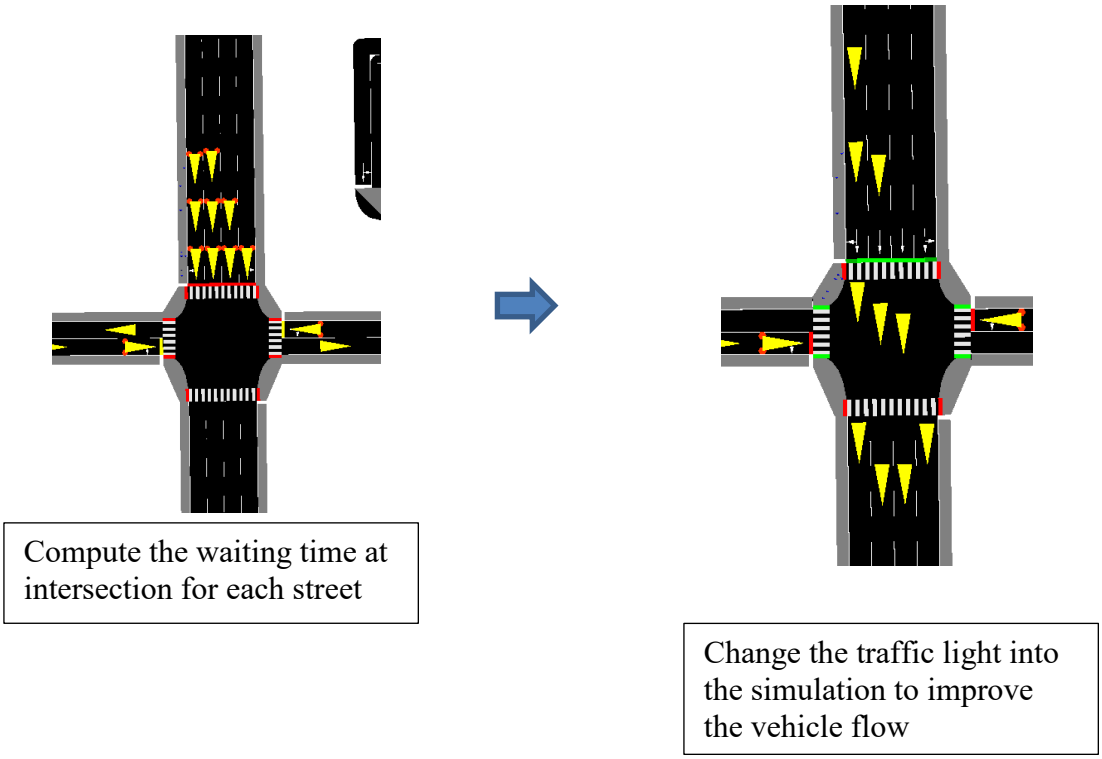


Figure 85: Simple optimization result

CHAPTER 7. CONCLUSION AND FUTURE WORK

Three main gaps were identified in the current method for ensuring pedestrian safety:

- There does not exist a means to conduct enforcement automatically using only surveillance cameras in a city environment.
- Surveillance videos cameras and existing tools on campus are not used to inform risk to pedestrian safety.
- No methods have been implemented that enable statistics of hazardous situation such that: jaywalking, red light running, speeding etc, using cameras only.

To answer these needs, a tool was built to extract data about high risk situations for pedestrians and vehicles around campus. The tool uses surveillance video cameras and stores information in a database.

The tool was tested on three different surveillance video cameras located at the intersection between Fifth Street NW and Spring Street NW, Fifth Street NW and Techwood Drive, and Ferst Drive and Atlantic Drive.

By using computer vision techniques and data fusion, incidents such as jaywalking, red light running, speeding, and situations with high risk of collision between vehicles were detected and stored in a database. Hence the first research question was answered and two hypotheses were validated.

7.1 Answers to Research Questions

The answer to the **Question 1:** "What approach(es) would allow for high-risk situations to be automatically detected?" was provided by validating the two following hypothesis:

- **Hypothesis 1.1**: If computer vision techniques are applied on traffic camera data then high-risk situations can be automatically detected.

Data useful for detecting high risk situations were extracted by using computer vision techniques such as persons and vehicles detection and tracking methods and color detection techniques. Then, the pedestrian and vehicle locations, stop line and crosswalk demarcations, and traffic light colors are input into a data fusion process that results in the automated detection of incidents. Hence, jaywalking, red light running, speeding, high risk of collision are determined in an automated way by this tool and **Hypothesis 1.1** is validated.

The method implemented provides additional information such as the vehicles and pedestrians count at intersection or the arriving and exiting flows per street. The count can be used by the Police department to schedule a police agent to manage traffic and pedestrian flows when needed. The flows can be used to calibrate simulations.

- **Hypothesis 1.2:** If thresholds are applied on speed, position, pedestrian-vehicle distance, deceleration during braking then critical traffic situations can be identified and classified.

Once the location information was extracted, the use of a threshold on the speed enables the determination of speeding and the use of a threshold on the location enables detection of jaywalking, red light running, and high risk of collision. Therefore **Hypothesis 1.2** was validated by combining thresholds with the information extracted from the computer vision techniques.

7.2 Benefits of the tool

All the information that is collected is stored into a database which helps to complete the current police record database that gathers incidents data around campus. The new data added to the database are in a structured format which enables the calculation of statistics easily by applying simple queries on the tables. The current Police record database contains a text description of the incidents which makes it more tedious to classify incidents as a collision, red light running, or other violations.

This tool could be used to enforce the law and penalize wrong behaviors in an automated way by sending citations when violations are detected (Figure 86). Enforcing the law automatically is expected to lead to better vehicle behavior and so to a safer environment for pedestrians.

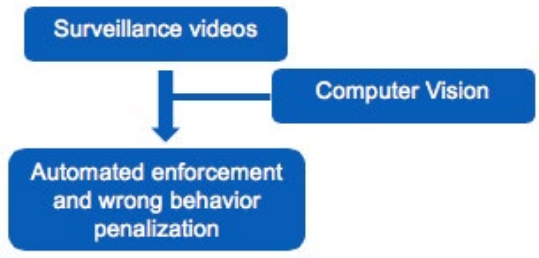


Figure 86: Automated law enforcement

If the tool is installed on the Georgia Tech Police Department server, then it could help spotters to better detect dangerous behaviors. It could consequently improve their response time if an important incident occurs. Moreover, this tool could enable the issuance of citations automatically which in turn enables a time savings for the Police department that can use its resources for other needs.

If the tool is implemented on the Georgia Tech Police Department server, then the Police will have access to additional statistics about pedestrian safety on intersections. These data can be used to evaluate the needs for enforcement or engineering measures on campus (Figure 87). Once these measures are implemented, pedestrian safety should be improved around campus.

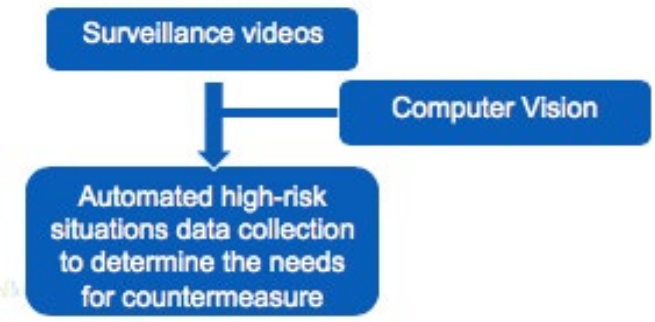


Figure 87: Automated data collection

7.3 Future Work

Improvement could be done to improve the tool:

- The green color detection of the lights:

The green color detection is not as accurate as the orange and red color detection. Detecting the brightness of a given light and not its color might be a solution. The color detection at night could be improved by using the same method or by modifying the range of red and orange pixel intensities used during the day.

- The speed determination:

  An assumption was made about the conversion from pixel to meter. The diagonal of the image length in meter and pixel were used to establish this relation.

  However for more accuracy, the relation should be done for pixels of the x-axis and pixels of then y-axis separately.

- The use of a second camera (webcam) would enable information about the third dimension and the ability to compute the surrogate safety factors. Once this information is obtained, the surrogate safety indicators such as the time to collision, the post encroachment time, or the gap time can be computed. These data will help to divide the traffic interactions as critical, dangerous, or safe. Once it is done, only critical and dangerous interactions would be selected and stored in the database to avoid a large increase in storage costs.
- Add additional incidents into the database such as pedestrian-pedestrians collisions
- Modify the TracI API to take pedestrians into account

Concerning this last improvement, more details are given below.

A current safety problem is the traffic light scheduling for motorized and non-motorized traffic between 5th street NW and Spring Street NW. The implementation of reinforcement learning strategies should lead to a more optimized traffic light system and so to a safer intersection as the jaywalking risk will be reduced (Figure 88).

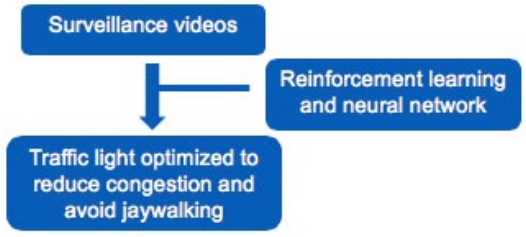


Figure 88: Traffic Light Control

The reinforcement learning algorithm must run in parallel to a simulation of the traffic at the intersection. The intersection was calibrated in SUMO using real data extracted from surveillance cameras. However, the implementation of this technique could not be achieved entirely due to that fact that retrieving pedestrian data while the simulation is running in SUMO has not been implemented into the TracI API yet.

In a future work, the source code of SUMO could be changed in order to add features about the TracI module for pedestrians. The first function to implement would return the real and accurate position of a pedestrian. Then this function can be used to determine the speed and when the speed is close to zero the number of pedestrian waiting can be inferred.

If this method is successful, it will help the Police to reduce the resources required to manage the traffic at this intersection. It could be used as well to manage traffic in an automated way when an athletic events occur. Then police agent resources can be used for other needs of the Police Department.

Moreover, in addition to decreasing the waiting time of pedestrians and so the risk of jaywalking, this method would avoid traffic congestions and reduce delays for vehicles. Because delays imply costs (especially for businesses), this method would enable better efficiency and a cost reduction.

APPENDIX A. SET UP THE VIRTUAL ENVIRONEMENT

How to set up an environment to use the dlib library and the tracking algorithm implemented during this thesis:

Create a virtual environment called for instance ‘env_dlib’:

Write the following command into the prompt:

conda create –name env_dlib-env python=3.6

To enter in this virtual environment in the prompt, print: *conda activate env_dlib*

To go into Python: write *python* in the prompt

To install a library that require this environment write

Pip install NameOfLibrary

Example

Pip install dlib

Then write *exit()* in the prompt when the user is done with the use of Python

Finally write *Conda deactivate* to leave the virtual environment

The virtual environment is store into a path that you choose, for instance “users:dcommun3:Anaconda:envs:env_dlib’ then use the bin or the python.exe executable of these path into the settings of Pycharm

REFERENCES

[1] NHTSA, "Traffic Safety Facts".

[2] U. C. bureau, "US population growth per year".

[3] J. C. Turner, . E. V. Leno and A. Keller, "Causes of Mortality Among American College Students: A Pilot Study," 2013.

[4] K. M. P. Pollack, A. C. Gieloen and M. Mitzner, "Investigating and Improving pedestrian safety in an urban environment," 2014.

[5] "Pedestrian Safety Enforcement Operations : a How-to-How guide ".

[6] "Georgia Tech Institutional Research and Planning".

[7] Statista, " Google Play : number of available apps 2009-2018".

[8] "United States mobile phone users 2012_2020," Statista.

[9] D. Stavrinos, K. W. W. Byington and David C., Distracted walking: Cell phones increase injury risk for college pedestrians, Journal of safety research, 2011.

[10] D. Gettman and L. Head, "Surrogate Safety Measures from Traffic Simulation Models," 2003.

[11] "Traffic calming device implementation Guidebook," city of Atlanta.

[12] Traffic Calming, Norwalk transportation management plan.

[13] "How to Improve Pedestrian Safety, A Guide for Communities".

[14] Automated enforcement overview, National Conference of State Legislatures.

[15] An Overview of Automated Enforcement Systems and Their Potential for Improving Pedestrian and Bicyclist Safety, Pedestrian and Bycicle Information Center.

[16] Automated Safety Enforcement: A critical tool to achieve Vision Zero. http://walksf.org/dev/wp-content/uploads/2014/02/Automated-Safety-Enforcement-Fact-Sheet-FINAL.pdf.

[17] A. V. Laura Jateikiene, "Average speed enforcement system efficiency assessment model," 2017.

[18] "Allied Vision," [Online]. Available: alliedvision.com.

[19] "Eyes on the street: How wireless video solutions are transforming public safety," Motorola.

[20] E. G. Learned-Miller, Introduction to Computer Vision, University of Massachussetts, Amherts, Department of Computer Science.

[21] H. Idrees, M. Shah and R. Surette, "Enhancing camera surveillance using computer vision: a research note," Center of Research in Computer Visions, University of Central Florida and Department of Criminal Justice, University of Central Florida, 2018.

[22] A. Kinoshitaa, . A. Takasub and J. Adach, "Real-time traffic incident detection using a probabilistictopic model," 2015.

[23] R. P Loce, E. Andres Bernal and R. Baia, "Computer vision in roadway transportation systems: A survey," 2013.

[24] "Company Traffic Vision," [Online]. Available: http://www.trafficvision.com/.

[25] B. Solano L, A. Pilonieta and . F. Guzm, "Video-based assessment of pedestrian behavior: Development and testing of methods.," 2017.

[26] P. Olszewski, . I. Buttler and W. Czajewski, Pedestrian safety assessment with video analysis, 2016.

[27] X. Wang, H. SonG and . H. Cui, Pedestrian abnormal event detection based on multi-feature fusion in traffic video, 2018.

[28] K. Shaaban and K. Abdel-Warith , "Agent-based Modeling of Pedestrian Behavior at an Unmarked Midblock Crossing. ," 2017.

[29] P. Chen, W. Z. Zeng, . Y. W. Wang and G. Yu, "Surrogate Safety Analysis of Pedestrian-Vehicle Conflict at Intersections Using Unmanned Aerial Vehicle Videos," 2017.

[30] T. Sayed, K. Ismail and N. Saunier, Automated Analysis of Pedestrian-Vehicle Conflicts Using Video Data, 2009.

[31] "Pedestrian Safety Evaluation: Quantitative Data Collection Methods".

[32] P. Olszewski, I. Buttler and W. Czajewski, Pedestrian safety assessment with video analysis, 2016.

[33] "Deep AI," [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/computer-vision.

[34] S. Achartz, "State of the art of object recognition techniques," 2017.

[35] w. l. v. s. i. s. IPVM, "The Source for Security and Video Surveillance Information".

[36] P. Read and M.-P. Meyer, "Restoration of Motion Picture Film," 2000.

[37] G. Zhong, "Object Detection and Tracking in Video," 2001.

[38] [. o. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/PCA.pdf], "PCA lecture," Auckland University.

[39] "CSE Lecture 6: Features and Image Matching," Washington University.

[40] "CS 126 Lecture T1: Pattern Matching," Princeton University.

[41] H. Frezza-Buet, . M. Geist and F. Penne, Machine learning, CentraleSupelec lecture, CentraleSupelec, 2018.

[42] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way," [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[43] A. Simon, "State of the Art of ObjectRecognition Techniques".

[44] "Caffe Deep learning framework," [Online]. Available: http://caffe.berkeleyvision.org.

[45] J. Johnson, "CS231n: Convolutional Neural Network for Visual Recognition," Stanford CS class.

[46] Y. Ni, M. Wang, J. Sum and . K. Li, Evaluation of pedestrian safety at intersections: a theoritical framework based on pedestrian-vehicle interaction pattern, 2016.

[47] Y. Ni, "Evaluation of pedestrian safety at intersections: a theoritical framework based on pedestrian)vehicle interaction patterns," 2016.

[48] J. Mead, C. Zegeer and M. Bushell, Evaluation of Pedestrian-Related Roadway Measures:A Summary of available research, 2014.

[49] Y. Liu, L. Liu and W.-P. Chen, Intelligent Traffic Light Control Using Distributed Multi-agent Q learning, 2017.

[50] Department of Transportation, United States. Adaptive signal control technology..

[51] D. o. transport, "Traffic Advisory Leaflet 4/95: The SCOOT Urban Traffic Control System," 1995.

[52] N. Government., " Roads and Traffic Authority. "SCATS - Product Features". SCATS.".

[53] Y. Zhao and Z. Tian, "An Overview of the Usage of Adaptive Signal Control System in the United States of America," 2012.

[54] I. Arel, T. Urbanik and C. Liu, Reinforcement learning-based multi-agent system for network traffic signal control, 2010.

[55] S. Araghi, A. Khosravi and M. Johnstone, "Intelligent Traffic Light Control of Isolated Intersections Using Machine Learning Methods," Centre for Intelligent Systels Research (CISR), 2013.

[56] N. J. Nilsson, Introduction to machine learning, Stanford University, CA 94305.

[57] F. H. A. US department of transportation, Traffic Signal Timing Manual, Chapter 4, Traffic signal Design.

[58] X. Liand, X. Du, G. Wang and Z. Han, "Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks," 2018.

[59] R. Akcelik, "Estimation of delays at traffic signlas for variable demand conditions," 1993.

[60] Y. Zhang, R. Su and K. Gao, "Traffic Light Sceduling for Pedestrians and Vehicles," 2017.

[61] M. Saidallah, A. E. F. EI Fegrgougui and Abdelb, "A Comparative Study of Urban Road Traffic Simulators," 2016.

[62] F. Porikli and A. Yilmaz, "Object detection and tracking," MITSUBISHI ELECTRIC RESEARCH LABORATORIES, January 2012.

[63] "Caffe framework," [Online]. Available: Caffe.berkleyvision.org.

[64] "OpenStreetMap," [Online].

[65] "Video object tracking - dlib," [Online].

[66] D. e. al, "Accurate Scale Estimation for Robust Visual Tracking," 2014.

[67] ASDL, "MOTIFE Grand Challenge," in *EAB 2019*.

[68] "Mathworks," [Online].

[69] "Python," [Online].

[70] R. Collins, "Camera Projection II, CSE486, Penn State University".

[71] "Stereo and 3D Vision, Lecture 16, Washington University".

[72] "SUMO simulation of Uran mobility," [Online].

[73] Gudwin and Ricardo R., "Urban Traffic Simulation with SUMO," 2016.

[74] "source:https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/," [Online].

[75] T. Fu, N. S. Saunier and . L. Miranda-Moreno, "A novel framework to evaluate pedestrian safety at non-signalized locations," 2018.

[76] T. Carlo, "3D Reconstruction with two Calibrated Cameras".

[77] Z. Fan, . Y. Wu and . W. Li, "Automatic Pavement Crack Detection Based on Strcutured Prediction with the Convolutional Neural Network," 2018.